blender pdfwikimanual

Blender wiki PDF Manual conversion by Marco Ardito

Details, info, download: http://amrc.altervista.org

Updated: 07/10/2014 from:

http://wiki.blender.org/index.php/Doc:2.6/Manual

Table of Content

Introduction

Blender 2.5 with a Big Buck Bunny scene open

Welcome to Blender! The Blender documentation consists of many parts: this user manual, a reference guide, tutorials, forums, and many other web resources. The first part of this manual will guide you through installing Blender, and optionally building Blender from source.

Blender has a powerful interface, highly optimized for 3D graphics production. The large number of buttons and menus might be a bit intimidating at first, but don't worry. After a bit of practice it will become familiar and intuitive.

It is highly recommended you read our section on The Interface carefully to get familiar with both the interface and with the conventions used in the documentation.

What is Blender?

Blender was first conceived in December 1993 and became a usable product in August 1994 as an integrated application that enables the creation of a diverse range of 2D and 3D content. Blender provides a broad spectrum of modeling, texturing, lighting, animation and video post-processing functionality in one package. Through its open architecture, Blender provides cross-platform interoperability, extensibility, an incredibly small footprint, and a tightly integrated workflow. Blender is one of the most popular Open Source 3D graphics applications in the world.

Aimed at media professionals and artists world-wide, Blender can be used to create 3D visualizations and still images, as well as broadcast- and cinema-quality videos, while the incorporation of a real-time 3D engine allows for the creation of 3D interactive content for stand-alone playback or video games.

Originally developed by the company 'Not a Number' (NaN), Blender has continued on as 'Free Software', with the source code available under the GNU GPL license. The Blender Foundation in the Netherlands coordinates its ongoing development.

Between 2008 and 2010, key parts of Blender were re-written to improve its functions, workflow and interface. The result of this work produced the version of the software known as Blender 2.5.

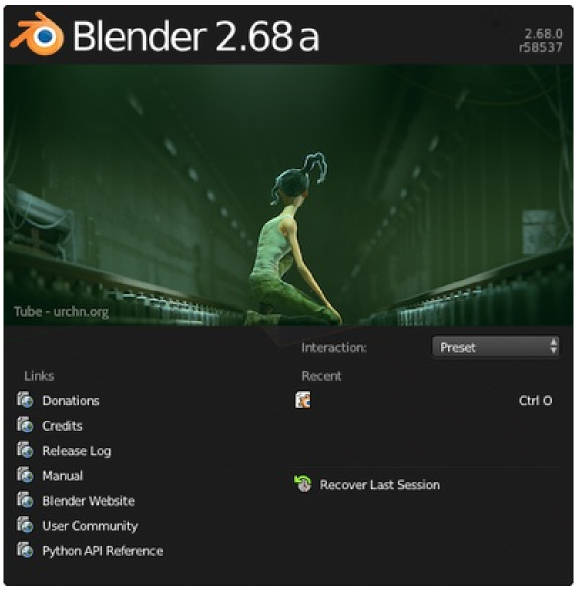
Key Features:

Image being rendered and post-processed

- Fully integrated creation suite, offering a broad range of essential tools for the creation of 3D content, including modeling, uv mapping, texturing, rigging, skinning, animation, particle and other simulation, scripting, rendering, compositing, post-production, and game creation;
- Cross platform, with an OpenGL GUI that is uniform on all platforms (customizable with python scripts), ready to use for all current versions of Windows (XP, Vista, 7), Linux, OS X, FreeBSD, Sun and numerous other operating systems;
- High quality 3D architecture enabling fast and efficient creation work-flow;
- More than 200,000 downloads of each release (users) worldwide;
- User community support by forums for questions, answers, and critique at http://BlenderArtists.org and news services at http://BlenderNation.com;
- Small executable size, easy distribution.

You can download the latest version of Blender here.

Blender is a full-featured tool



Blender Welcome Screen ver 2.68 autumn 2013

Blender makes it possible to perform a wide range of 3d-content-creation-oriented tasks. Therefore it may seem daunting when first trying to grasp the basics. However, with a bit of motivation and the right learning material, it is possible to be productive with Blender after a few hours of practice. If you're reading this wiki, it is a good start, though it serves more as a reference. You also have online video tutorials (free and paid) from specialized websites, and several books in the Blender store.

Despite everything Blender can do, it remains a tool. Great artists create masterpieces, not only by pressing buttons or manipulating brushes, but also by learning and practicing human anatomy, color theory, composition, lighting, traditional animation, photography, psychology and many other areas. 3D content creation software have the added technical complexity and jargon associated with the underpinning technologies. CPUs, GPUs, memory, algorithms, vectors, materials, meshes are the mediums of the digital artist, and understanding them, even broadly, will help you using Blender to its best.

So keep reading this wiki, learn the great tool that Blender is, keep your mind open to other artistic and technological areas, and you too can become a great artist.

Blender's History

In 1988 Ton Roosendaal co-founded the Dutch animation studio *NeoGeo*. NeoGeo quickly became the largest 3D animation studio in the Netherlands and one of the leading animation houses in Europe. NeoGeo created award-winning productions (European Corporate Video Awards 1993 and 1995) for large corporate clients such as multi-national electronics company Philips. Within NeoGeo Ton was responsible for both art direction and internal software development. After careful deliberation Ton decided that the current in-house 3D tool set for NeoGeo was too old and cumbersome to maintain and upgrade and needed to be rewritten from scratch. In 1995 this rewrite began and was destined to become the 3D software creation we all know as *Blender*. As NeoGeo continued to refine and improve Blender it became apparent to Ton that Blender could be used as a tool for other artists outside of NeoGeo.

In 1998, Ton decided to found a new company called Not a Number (NaN) as a spin-off of NeoGeo to further market and develop Blender. At the core of NaN was a desire to create and distribute a compact, cross platform 3D application for free. At the time this was a revolutionary concept as most commercial modelers cost several thousands of (US) dollars. NaN hoped to bring professional level 3D modeling and animation tools within the reach of the general computing public. NaN's business model involved providing commercial products and services around Blender. In 1999 NaN attended its first Siggraph conference in an effort to more widely promote Blender. Blender's first Siggraph convention was a huge success and gathered a tremendous amount of interest from both the press and attendees. Blender was a hit and its huge potential confirmed!

Following the success of the Siggraph conference in early 2000, NaN secured financing of €4.5m from venture capitalists. This large inflow of cash enabled NaN to rapidly expand its operations. Soon NaN boasted as many as fifty employees working around the world trying to improve and promote Blender. In the summer of 2000, Blender v2.0 was released. This version of Blender added the integration of a game engine to the 3D application. By the end of 2000, the number of users registered on the NaN website surpassed 250,000.

Unfortunately, NaN's ambitions and opportunities didn't match the company's capabilities and the market realities of the time. This over-extension resulted in restarting NaN with new investor funding and a smaller company in April 2001. Six months later NaN's first commercial software product, *Blender Publisher* was launched. This product was targeted at the emerging market of interactive web-based 3D media. Due to disappointing sales and the ongoing difficult economic climate, the new investors decided to shut down all NaN operations. The shutdown also included discontinuing the development of Blender. Although there were clearly shortcomings in the then current version of Blender, such as a complex internal software architecture, unfinished features and a non-standard way of providing the GUI, the enthusiastic support from the user community and customers who had purchased Blender Publisher in the past meant that Ton couldn't justify leaving Blender to fade into insignificance. Since restarting a company with a sufficiently large team of developers wasn't feasible, Ton Roosendaal founded the non-profit organization *Blender Foundation* in March 2002.

The Blender Foundation's primary goal was to find a way to continue developing and promoting Blender as a community-based Open Source project. In July 2002, Ton managed to get the NaN investors to agree to a unique Blender Foundation plan to attempt to release Blender as open source. With an enthusiastic group of volunteers, among them several ex-NaN employees, a fund raising campaign was launched to "Free Blender". To everyone's surprise and delight the campaign reached the €100,000 goal in only seven short weeks. Thence the Foundation could buy the rights to the Blender source code and intellectual property rights from the NaN investors and subsequently release Blender to the open source community under the terms of the GNU General Public License (GPL) on Sunday October 13, 2002. Blender development continues to this day driven by a team of far-flung, dedicated volunteers from around the world led by Blender's original creator, Ton Roosendaal.

Video: From Blender 1.60 to 2.50

[video link]

Version/Revision Milestones

Release Notes

To see release notes for each version, you can click on the version number.

- From 1.00 to 2.30, there are no more links for release notes;
- From 2.30 to 2.40, you can find release notes only at Blender Release Notes;
- From version 2.40 and up, the release notes are located at the Developers space in our wiki and at the Blender Release Notes

Blender's history and road-map:

The start !

- 1.00 – January 1995: Blender in development at animation studio NeoGeo.
- 1.23 – January 1998: SGI version published on the web, IrisGL.
- 1.30 – April 1998: Linux and FreeBSD version, port to OpenGL and X11.
- 1.3x – June 1998: NaN founded.
- 1.4x – September 1998: Sun and Linux Alpha version released.
- 1.50 – November 1998: First Manual published.
- 1.60 – April 1999: C-key (new features behind a lock, $95), Windows version released.
- 1.6x – June 1999: BeOS and PPC version released.
- 1.80 – June 2000: End of C-key, Blender full freeware again.

Blender 2.0

- 2.00 – August 2000: Interactive 3D and real-time engine.
- 2.10 – December 2000: New engine, physics, and Python.
- 2.20 – August 2001: Character animation system.
- 2.21 – October 2001: Blender Publisher launch.
- 2.2x – December 2001: Mac OSX version.

Blender goes Open Source

- **13 October 2002: Blender goes Open Source, 1st Blender Conference**.
- 2.25 – October 2002: Blender Publisher becomes freely available.
- Tuhopuu1 – Oct 2002: The experimental tree of Blender is created, a coder's playground.
- 2.26 – February 2003: The first true Open Source Blender.
- 2.27 – May 2003: The second Open Source Blender.
- 2.28x – July 2003: First of the 2.28x series.
- 2.30 – October 2003: Preview release of the 2.3x UI makeover presented at the 2nd Blender Conference.
- 2.31 – December 2003: Upgrade to stable 2.3x UI project.
- 2.32 – January 2004: Major overhaul of internal rendering capabilities.
- 2.33 – April 2004: Game Engine returns, ambient occlusion, new procedural textures.
- 2.34 – August 2004: Big improvements: particle interactions, LSCM UV mapping, functional YafRay integration, weighted creases in subdivision surfaces, ramp shaders, full OSA, and many many more.
- 2.35 – November 2004: Another version full of improvements: object hooks, curve deforms and curve tapers, particle duplicators and much more.
- 2.36 – December 2004: A stabilization version, much work behind the scene, normal and displacement mapping improvements.

A Big Leap

- 2.37 – June 2005: A big leap: transformation tools and widgets, softbodies, force fields, deflections, incremental subdivision surfaces, transparent shadows, and multithreaded rendering.
- 2.40 – December 2005: An even bigger leap: full rework of armature system, shape keys, fur with particles, fluids and rigid bodies.
- 2.41 – January 2006: Lots of fixes, and some game engine features.
- 2.42 – July 2006: The **Node** release. Over 50 developers contributed nodes, array modifier, vector blur, new physics engine, rendering, lipsync and, many other features. This was the release following Project Orange.
- 2.43 – February 2007: The **Multi** release: multi-resolution meshes, multi-layer UV textures, multi-layer images and multi-pass rendering and baking, sculpting, retopology, multiple additional matte, distort and filter nodes, modeling and animation improvements, better painting with multiple brushes, fluid particles, proxy objects, sequencer rewrite, and post-production UV texturing. whew! Oh, and a website rewrite. And yes, it still has multi-threaded rendering for multi-core CPUs. With Verse it is multi-user, allowing multiple artists to work on the same scene collaboratively. Lastly, render farms still provide multi-workstation distributed rendering.
- 2.44 – May 2007: The **SSS** release: the big news, in addition to two new modifiers and re-awakening the 64-bit OS support, was the addition of subsurface scattering, which simulates light scattering beneath the surface of organic and soft objects.
- 2.45 – September 2007: Another bugfix release: serious bugfixes, with some performance issues addressed.
- 2.46 – May 2008: The **Peach** release was the result of a huge effort of over 70 developers providing enhancements to the core and patches to provide hair and fur, a new particle system, enhanced image browsing, cloth, a seamless and non-intrusive physics cache, rendering improvements in reflections, AO, and render baking; a mesh deform modifier for muscles and such, better animation support via armature tools and drawing, skinning, constraints and a colorful Action Editor, and much more. It was the release following Project Peach.
- 2.47 – August 2008: Bugfix release.
- 2.48 – October 2008: The **Apricot** release: cool GLSL shaders, lights and GE improvements, snap, sky simulator, shrinkwrap modifier, python editing improvements.
- 2.49 – June 2009: The **Pre-Re-Factor** release added significant enhancements to the core and GE. Core enhancements include node-based textures, armature sketching (called Etch-a-Ton), boolean mesh operation improvements, JPEG2000 support, projection painting for direct transfer of images to models, and a significant Python script catalog. GE enhancements included video textures, where you can play movies in-game (!), upgrades to the Bullet physics engine, dome (fish-eye) rendering, and more API GE calls made available.

Blender 2.5 - The Recode !

- 2.5x – From 2009 to August 2011. This series release 4 pre-version (from Alpha0 - November 2009 - to Beta July 2010) and three stable versions (from 2.57 - April 2011 - to 2.59 - August 2011). It is one of the most important development project of blender with a total re-coding of the software with new functions, redesign of internal window manager and event/tool/data handling system, new python API... The final version of this project was Blender 2.59 in August 2011.
- 2.60 – October 2011: Internationalization of the UI, 3D Audio and Video. This release incorporates improvements in Animation System and Game Engine, Vertex Weight Groups Modifiers, 3D Audio and Video, Bug Fixes, and the UI Internationalization (Garlic Branch merged into trunk).
- 2.61 – December 2011: Camera Track, Ocean Simulation, Cycles Render Engine, Dynamic Paint. The new Cycles Render Engine is now added in the Blender default installation, also Camera Tracking for mixing footages with 3D, Dynamic Paint for modifying Textures with Mesh contact/approximation, the Ocean Simulation is a new Modifier to simulate Ocean and Foam (Ported from the open source Houdini Ocean Toolkit), New Addons, Bug Fixes, and more extensions added for the Python API.
- 2.62 - February 2012: Carve Booleans, Motion Tracking, Remesh Modifier. The Carve library is now added to improve results when performing Boolean operations, Blender now support Motion Tracking for object movements in the Scene, the Remesh Modifier generate new topology using an input Mesh as a base, many improvements in Game Engine, Collada, Bump Mapping, Dynamic Paint, UV Tools, Cycles Render Engine, Matrices and Vectors in Python API were improved, New Addons, and many bugs were fixed.

2.63 - Bmesh - Blender with N-gons

- 2.63 - April 2012: A new mesh system has been added to Blender, with full support for N-sided Polygons instead of only triangles and quads , Sculpt Hiding, Cycles Render with panoramic Camera, mirror ball environment textures and float precision textures, render layer mask layers, ambient occlusion and viewport display of background images and render layers, Motion Tracker with few smaller improvements, new Import and Export Addons were added, and Renderfarm.fi now supports Cycles. 150 bugfixes for bugs that existed in previous releases.

2.64 - The Open Source VFX release

- 2.64 - October 2012: Mask Editor, Improved Motion Tracker, Opencolor IO, Cycles Render improvements, Sequencer improvements, better Mesh Tools (Inset and Bevel were improved), new Compositing Nodes for Green Screen, Sculpt Masking, Collada improvements for Game Engines, New Skin Modifier, new compositing Nodes Backend, and many bugs were fixed.

2.65 - Continuous Improvements

- 2.65 - December 2012: Fire and Smoke, Anisotropic shaders for Cycles , Modifier improvements, Bevel tool now includes rounding, new Addons, and more than 200 bugs that existed in previous versions have been fixed, resulting on a **2.65a** release!

2.66 - Dynamic Topology, Rigid Body Simulation

- 2.66 - February 2013: Dynamic Topology Sculpting, Rigid Body Simulation, improvements in UI and usability (including Mac new 'Retina Display' support), Cycles Render now supports hair, Improvements in image transparency, the bevel tool now supports individual vertex bevelling, new Mesh Cache Modifier and the new UV Warp Modifier, a new SPH particle fluid solver was added to calculate fluid dynamics, improvements in game engine and collada, support for vertex colors bake, more efficient ambient occlusion baking for multires meshes, edge based UV stitching, more control over mapping texture brushes for texture painting, gradient tools for weight painting, and a translate node for the compositor. A New Addon for MilkShape 3D format support and EDL Video Import. More than 250 bugs that existed in previous versions have been fixed, resulting on a **2.66a** release!

2.67 - Freestyle, 3d printing

- 2.67 - May 2013: Freestyle non-physical line rendering engine, paint system improvements, Subsurface scattering, Ceres library in Motion Tracker, border in Compositing Nodes Viewer, new custom python nodes, multiple independent node editors, nested node groups, new mesh modelling tools - inset and poke face, knife tool, better support for UTF8 text and improvements in text editors, new add-ons for 3d printing, node efficiency tools and VRML2 support.

2.68 - Continuous Improvements

- 2.68 - July 2013: New and improved modelling tools: Rewritten bridge tool, grid fill, improvements to proportional editing mode, snap to symmetry, dissolve, vertex connect, Cycles Rendering improved with three new nodes: Wavelength, Toon BSDF, Wireframe node, and with new render passes and changes in ray visibility, new closures in Open Shading Language added, big improvements in Motion Tracker (reconstructed scene ambiguity, added scene orientation and refining markers position, added automatic keyframe selection), physics improvements: added the ability to generate particles on meshes changed by stack of modifiers, new options added to smoke simulations (subframes and full sampling), improved usability, Python Security, two new addons added, and over 280 bugfixes.

2.69 - Continuous Improvements

- 2.69 - October 2013: New and improved modelling tools: Hidden Wire Display for retopology, Bridge, Edgenet Fill, Bisect, Grid Fill, Symmetrize, Curve and Lattice editing tools, Cycles Rendering improved in many areas: bumpmapping for SSS, Branched Path Trace Integrator is available for CPU, Hosek/Wilkie Sky model, new nodes for Cycles: Hair BSDF, Ray Depth, Blackbody, Vector Transform, Combine/Separate HSV, new options for Mapping node, improved usability of Cycles UI, new additions to tone mapping, Plane Tracking added to Motion Tracker, numerous small features were added with improvements for vertex parenting, constrains, mask editing, texture painting, animation, empty objects, images, UI lists, viewport roll, BGE, addons, better support for FBX import/export, and over 270 bugs fixed.

About Free Software and the GPL

When one hears about "free software", the first thing that comes to mind might be "no cost". While this is true in most cases, the term "free software" as used by the Free Software Foundation (originators of the GNU Project and creators of the GNU General Public License) is intended to mean "free as in freedom" rather than the "no cost" sense (which is usually referred to as "free as in free beer"). Free software in this sense is software which you are free to use, copy, modify, redistribute, with no limit. Contrast this with the licensing of most commercial software packages, where you are allowed to load the software on a single computer, are allowed to make no copies, and never see the source code. Free software allows incredible freedom to the end user. Since the source code is universally available, there are also many more chances for bugs to be caught and fixed.

When a program is licensed under the GNU General Public License (the GPL):

- you have the right to use the program for any purpose;
- you have the right to modify the program, and have access to the source codes;
- you have the right to copy and distribute the program;
- you have the right to improve the program, and release your own versions.

In return for these rights, you have some responsibilities if you distribute a GPL'd program, responsibilities that are designed to protect your freedoms and the freedoms of others:

- You must provide a copy of the GPL with the program, so that recipienta are aware of their rights under the license.
- You must include the source code or make the source code freely available.
- If you modify the code and distribute the modified version, you must license your modifications under the GPL and make the source code of your changes available. (You may not use GPL'd code as part of a proprietary program.)
- You may not restrict the licensing of the program beyond the terms of the GPL. (You may not turn a GPL'd program into a proprietary product.)

For more on the GPL, check the GNU Project Web site.

Note: The GPL only applies to the blender application and **not** the artwork you create with it; for more info see: Blender License

Getting support: the Blender community

Being freely available from the start, even while closed source, helped considerably in Blender's adoption. A large, stable and active community of users has gathered around Blender since 1998. The community showed its support for Blender in 2002 when they helped raise €100,000 in seven weeks to enable Blender to go Open Source under the GNU GPL.

The community spans two widely overlapping sites:

The Development Community, centered around the Blender Foundation site. Here you will find the home of the development projects, the Functionality and Documentation Boards, the CVS repository with Blender sources, all documentation sources, and related public discussion forums. Developers contributing code to Blender itself, Python scripters, documentation writers, and anyone working for Blender development in general can be found here.

Go to http://www.blender.org

The User Community, centered around the independent BlenderArtists site. Here Blender artists, gamemakers and fans gather to show their creations, get feedback, ask for and offer help to get a better insight into Blender's functionality. Blender Tutorials and the Knowledge Base can be found here as well.

Go to http://www.BlenderArtists.org

These two websites are not the only Blender resources. The worldwide community has created a large number of independent sites, in local languages or devoted to specialized topics. A constantly updated list of Blender resources can be found at the above mentioned sites.

IRC chat channels

For immediate online feedback there are three IRC chat channels permanently open on irc.freenode.net. You can join these with your favorite IRC client:

- #blender Community support channel
- #blenderchat for general discussion of blender
- #blenderqa for asking questions on Blender usage
- #gameblender for discussion on issues related to game creation with Blenders included game engine

For developers there is also :

- #blendercoders for developers to ask questions and discuss development issues, as well as a meeting each Sunday at 4 pm Netherlands time
- #blenderpython for discussion of the python API and script development
- #blenderwiki for questions related to editing the wiki

Who uses Blender?

The Blender community is made up of people from all over the world, with novice and professional graphic artists, occasional users and commercial houses. This manual is written to serve the wide array of Blender users. You might be interested in using Blender if you are a:

- Hobbyist/Student that wants to explore the world of computer graphics (CG) and 3D animation.
- 2-D artist that produces single image art/posters or enhances single images as part of an image post-processing lab.
- 2-D artist or team that produces cartoon/caricature animations for television commercials or shorts (such as “The Magic of Amelia”).
- 3-D artist that works alone or with another person to produce short CG animations, possibly featuring some live action (such as "Suburban Plight").
- 3-D team that produces an animated (100% CG) movie (such as "Elephant's Dream", "Plumiferos", "Big Buck Bunny" or "Sintel").
- 3-D team that works together to produce live action movies that include some CG.

2D and 3D teams that produce movies and animations often specialize in certain aspects of CG. Some of these specific jobs that could use Blender include:

- Director - Defines what each Scene should contain, the action (animation) and shots (camera takes) within that scene.
- Modeler - Makes assets for the production. Specialties include Character, Prop and Landscapes/Stage modelers.
- Cameraman, Director of Photography (DP) - sets up the camera and its motion, shoots the live action, renders the output frames.
- Material Painter - paints the set, the actors, and anything that moves.
- Animation and Rigging - makes things hop about using armatures.
- Lighting and Color Specialist - Lights the stage and sets, adjusts colors to look good in the light, adds dust and dirt to materials, scenes, and textures.
- Special Purpose talent - Fluids, motion capture, cloth, dust, dirt, fire and explosion.
- Editor - takes all the raw footage from the DP and sequences it into an enjoyable movie.

Introduction

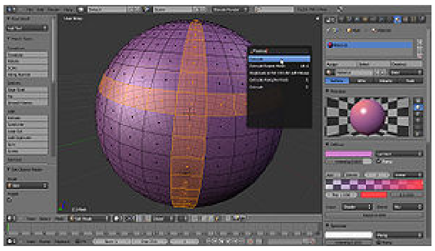
With version 2.5, Blender has seen phenomenal improvements in virtually all areas: software, interface, modeling, animation flow, tools, the python API, etc. This is the result of a careful study of use cases, years of additions and community collaboration, and a complete reorganization and rewrite of the software source code. As a result, this is one of the largest projects undertaken on the Blender code base to date.

This page explains the most striking differences between Blender 2.4 and Blender 2.5. This is not an exhaustive list of new functionality (that would be too long!) but is rather a concise introduction to the evolution of 2.5 and its major improvements over previous versions.

[video link]

Interface

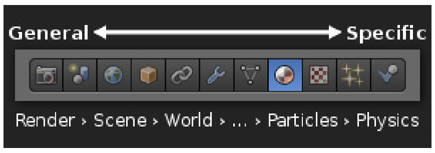
New User Interface



The Blender User Interface is based on 3 principles:

1. **Non Overlapping** : The UI permits you to view all relevant options and tools at a glance without pushing or dragging windows around.
2. **Non Blocking** : Tools and interface options do not block the user from any other parts of Blender. Blender doesn't pop up requesters that require the user to fill in data before things execute.
3. **Non Modal** : User input should remain as consistent and predictable as possible without changing commonly used methods (mouse, keyboard) on the fly.

The User Interface has been reorganized. Old *Buttons Windows* are now ***Properties***. Properties present data to users. Everything you see in the Properties can be animated, driven, and freely changed by the user. This means there are no tools here. These go to the new ***Toolbar*** of the different editors (like 3D view).



Starting at the top level, the Properties editor contains a list of tabs. The list of tabs themselves are organized so that the most general controls appear on the left (Render Properties), while more fine-grained controls (Object>Mesh>Material>Texture) appear on the right, following reading direction. Furthermore, available tabs depend on the selection (i.e. Mesh options are different from Camera options).

Read more about new UI design rules »

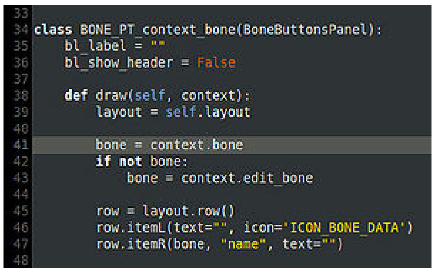
Read more about 2.5 UI Paradigms »

Read more about new properties panel »

Multi-screen

With its new Window Manager, Blender allows configuration of multiple windows/screens which is useful for multi-screen setups. As with the main window, each new window can be subdivided into areas.

Customizable

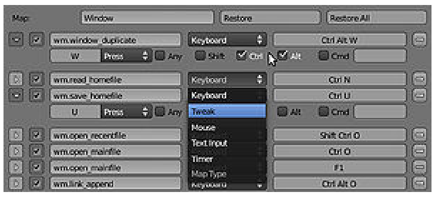


```
class BONE_PT_context_bone(BoneButtonsPanel):
    bl_label = ""
    bl_show_header = False

    def draw(self, context):
        layout = self.layout

        bone = context.bone
        if not bone:
            bone = context.edit_bone

        row = layout.row()
        row.itemL(text="", icon='ICON_BONE_DATA')
        row.itemR(bone, "name", text="")
```

The UI is more flexible than it was in 2.4x. Thanks to the new python API, it is possible to customize the interface and change the place of panels or buttons. Most of the interface uses python scripts available in the /.blender/scripts/ui/ folder so you can edit them easily and make your own Blender interface.

Thanks to this new python API, it is easier for the developer to integrate scripts in the Blender interface (like render engine, tools, import/export scripts...).

Read more about new python API »



Furthermore, Blender 2.5 includes a new ***Keymap Editor***. Hotkey/mouse definitions are grouped together in 'key maps'. For each editor in Blender as well as for all modes or modal tools like transform, there are multiple key maps. Customizing the keys is done by making a local copy of the default map and then editing all the options you'd like to have. The default key maps will always be unaltered and available to use.

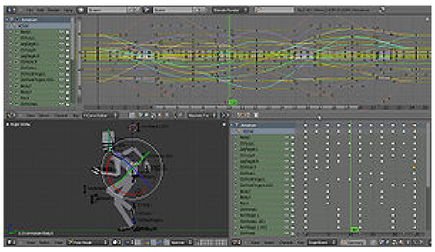
Animation system

Everything is animatable!

In Blender 2.5 every property can be animated, from the output image size to the modifiers options. Now you can set keys in every editor: 3D view, video sequence editor, Node editor (material, texture, composite)... This new system is called *Animato*.

Read more about Animato »

Dope sheet and graph editor



The IPO Curves Editor, Action Editor, and NLA Editor have been rebuilt into the ***Dope Sheet*** and ***Graph Editor*** (generic name used also in Maya).

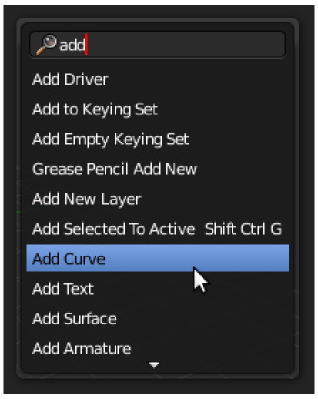
The "Action Editor" has been extended to become a full Dope Sheet, allowing control over multiple actions at once, grouping per type, and with better access to shape keys.

Blender's new animation system also allows the addition of a Function Curve to any property. The new Graph Editor (formerly Ipo Curve Editor) enables viewing, browsing and editing of any collection of function curves, including all the curves of an entire scene!

Watch this character animation »

New functions

Search tool

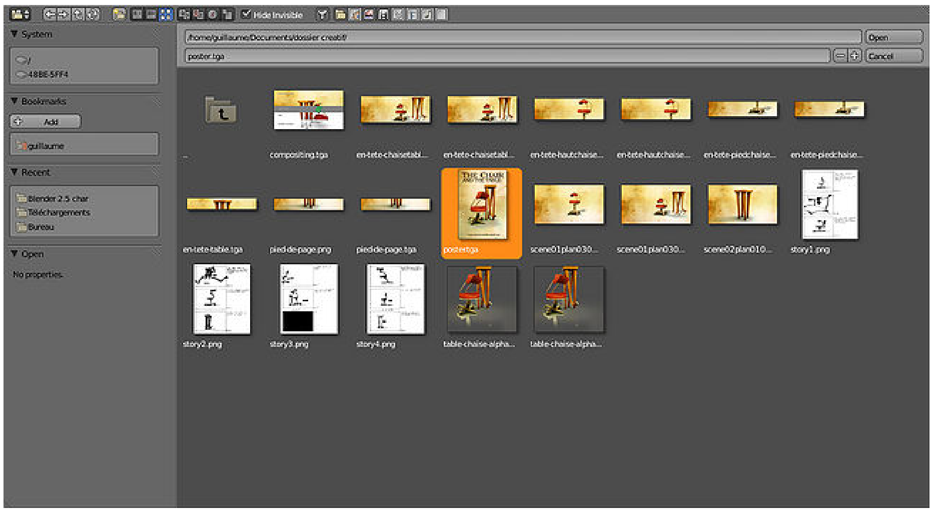


Blender 2.5 integrates a search tool which permits you to find a function by entering its name (or a part of it). Just hit Space where you want to search and the menu will appear. It is also available at the top of the Blender screen.

File browser improvements

The old file browser and Image browser have been linked into a single powerful browser. Files can be displayed as lists or thumbnails, and a new filter permits selection of which file types you want to show in the browser.

A side bar has also been added where you can see your disks, the most recent folder used, and a new function lets you create bookmarks !



Python API

Now based on Python 3.2

Watch this page on video!

This page has been made into a video. You can watch it on YouTube!

[video link]

About this manual

This manual is a mediawiki implementation that is written by a world-wide collaboration of volunteer authors. It is updated daily, and this is the English version. Other language versions are translated, generally, from this English source for the convenience of our world-wide audience. It is constantly out of date, thanks to the tireless work of some 50 or more volunteer developers, working from around the world on this code base. However, it is the constructive goal to provide you with the best possible professional documentation on this incredible package.

To assist you in the best and most efficient way possible, this manual is organized according to the creative process generally followed by 3D artists, with appropriate stops along the way to let you know how to navigate your way in this strange territory with a new and deceptively complex software package. If you read the manual linearly, you will follow the path most artists use in both learning Blender *and* developing fully animated productions:

1. Getting to know Blender = Intro, Navigating in 3d, scene mgt
2. Models = Modeling, Modifiers
3. Lighting
4. Shading = Materials, Textures, Painting, Worlds & Backgrounds
5. Animation = Basics, Characters, Advanced, Effects & Physical Sim
6. Rendering = Rendering, Compositing, Video Seq Edit
7. Beyond Blender = Extending Blender

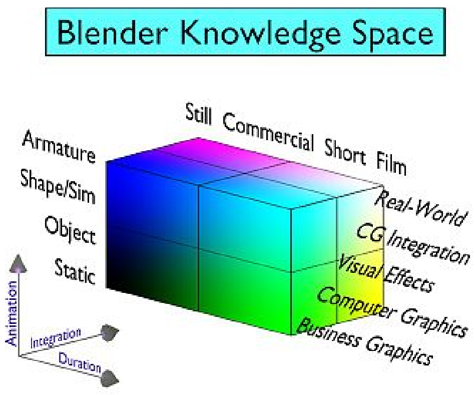
Audience

This manual is written for a very broad audience, to answer the question "I want to *do something*; how do I do it using Blender?" all the way to "what is the latest change in the way to sculpt a mesh?"

This manual is a worldwide collaborative effort using time donated to the cause. While there may be some lag between key features being implemented and their documentation, we do strive to keep it as up-to-date as possible. We try to keep it narrowly focused on what you, the end user, need to know, and not digress too far off topic, as in discussing the meaning of life.

There are other Blender wiki books that delve deeper into other topics and present Blender from different viewpoints, such as the Tutorials, the Reference Manual, the software itself, and its scripting language. So, if a question is not answered for you in this User Manual, please search the other Blender wiki books.

Learning CG and Blender



Getting to know Blender and learning Computer Graphics (CG) are two different topics. Learning what a computer model is, and then learning how to develop one in Blender are two different things to learn. Learning good lighting techniques, and then learning about the different kinds of lamps in Blender are two different topics. The first, or conceptual understanding, is learned by taking secondary and college courses in art and media, by reading books available from the library or bookstore on art and computer graphics, and by trial and error. Even though a book or article may use a different package (like Max or Maya) as its tool, it may still be valuable because it conveys the concept.

Once you have the conceptual knowledge, you can easily learn Blender (or any other CG package). Learning both at the same time is difficult, since you are dealing with two issues. The reason for writing this is to make you aware of this dilemma, and how this manual attempts to address both topics in one wiki book. The conceptual knowledge is usually addressed in a short paragraph or two at the beginning of a topic or chapter, that explains the topic and provides a workflow, or process, for accomplishing the task. The rest of the manual section addresses the specific capabilities and features of Blender. The user manual cannot give you the full conceptual knowledge - that comes from reading books, magazines, tutorials and sometimes a lifetime of effort. You can use Blender to produce a full-length feature film, but reading this manual and using Blender won't make you another Steven Spielberg!

At a very high level, using Blender can be thought of as knowing how to accomplish imagery within three dimensions of activity:

1. Integration - rendering computer graphics, working with real-world video, or mixing the two (CGI and VFX)
2. Animation - posing and making things change shape, either manually or using simulation
3. Duration - producing a still image, a short video, a minute-long commercial, a ten minute indie short, or a full-length feature film.

Skills, like navigating in 3D space, modeling, lighting, shading, compositing, and so forth are needed to be productive in any given area within the space. Proficiency in a skill makes you productive. Tools within Blender have applicability within the space as well. For example, the video sequence editor (VSE) has very little to do with the skill of animation, but is deeply applicable along the Duration and Integration scales. From a skills-learning integration perspective, it is interesting to note that the animation curve, called an Ipo curve, is used in the VSE to animate effects strips.

At the corners/intersections is where most people's interest's lie at any given time; a sort of destination, if you will. For example, there are many talented artists that produce Static-Still-CG images. Tony Mullen's book *Introducing Character Animation With Blender* addresses using CG models deformed by Armatures and shapes to produce a one-minute animation. Using Blender fluids in a TV production/commercial is at the Shape/Sim-Integrated-Minute intersection. Elephants Dream and Big Buck Bunny is a bubble at the Armature-CG-Indie space. Therefore, depending on what you want to do, various tools and topics within Blender will be of more or less interest to you.

A fourth dimension is Game Design, because it takes all of this knowledge and wraps Gaming around it as well. A game not only has a one-minute cinematic in it, but it also has actual game play, story line programming, etc. -- which may explain why it is so hard to make a game; you have to understand all this stuff before you actually can construct a game. Therefore, this Manual does not address using the Game Engine; that is a whole 'nother wiki book.

Downloadable versions of this manual

While the official documentation is created and maintained in this wiki, which always contains the latest contributions, it could be useful for someone to have a downloadable form of the wiki manual content (such as PDF format or others).

To address this need, users started creating **unofficial** versions of the wiki manual content:

- Unofficial conversion of the wiki manual, in PDF format
  - available from this page: https://archive.org/details/BlenderWikiPDFManual
  - more details available from the download page
  - update expected every month
  - last update: 2014-05-06
  - PDF size: ~44MB, ~1500 pages (english wiki version)
  - other languages: JA, ET, RO, DE, ES, CZ

Installing the Binaries

Blender 2.71 is available both as a binary executable and as source code on the Foundation site (http://www.blender.org/). Currently, to download Blender 2.71, select "Download Blender" from the right hand navigation menu on the homepage .

For the online manual hosted at the wiki, you can generally use the most recent version of Blender located at the Blender Foundation website (although all of the features from the newest release version may not be fully updated). If you are using a published version of this manual it is recommended that you use the Blender version included on the Guide CD-ROM. In the following text, whenever "download" is mentioned, those using the book should instead retrieve Blender from the CD-ROM.

Downloading and installing the binary distribution

Binary distributions are provided for the following operating system families:

- Windows
- Linux
- MacOSX
- FreeBSD

Some unofficial distributions may exist for other operating systems, but as they're not supported by the Blender Foundation, you should report any issues you may have with them directly to their maintainers.

Binaries for the Macintosh operating systems are provided for two different hardware architectures (x86 for Intel and AMD processors, and PowerPC), and for the choice between statically linked or dynamically loaded libraries.

The installer will create files and several folders in two locations on your computer: one set of folders is for the Blender program, and the other is a set of folders for your user data. You must have administrator authorization to create these. The folders are:

- .blender - configuration information (mostly prompts in your native language)
- blendcache_.B - temporary space for physics simulation information (softbodies, cloth, fluids)
- scripts - python scripts that extend Blender functionality
- tmp - temporary output, intermediate renders

Hardware Support

Blender supports 64-bit hardware platforms, removing the 2GB addressable memory limit.

Blender also supports multi-CPU/core chips such as the Intel Core-Duo and AMD X2 chips. A Threads setting is provided in the performance section of the render options to indicate how many cores to use in parallel when rendering. The Auto-detect setting will utilize all the cores available on your system, while the Fixed setting allows the user to manually specify the number of cores to be used when rendering.

Blender supports a wide variety of pen-based tablets on all major operating systems, in particular OS X, Windows XP, and Linux OSes.

Information on how to make render times shorter can be found in the Render section of the manual.

Developers platforms

This is the list of systems in use and supported by active Blender developers:

Name	OS	CPU	Graphics card
Andrea Weikert	Windows XP 32	AMD Athlon 64 X2	Nvidia Quadro FX1500
Andrea Weikert	Windows XP 32	Intel P4	ATI Radeon 9000
Antony Riakiotakis	Windows 7 64	Intel Core i5	NVidia Geforce GT 540M
Antony Riakiotakis	Ubuntu 12.10	Intel Core i5	NVidia Geforce GT 540M
Bastien Montagne	Debian Testing 64	Intel Core i7 Q720	ATI Radeon 5730 mobility
Benoit Bolsee	Windows XP 32	AMD Athlon XP	ATI Radeon 9200
Brecht van Lommel	Windows 7 64	Intel Core 2 Duo	NVidia GeForce 460 GTX
Daniel Genrich	Windows Vista 64	Intel Core 2 Duo	NVidia GeForce 8500 GT
Joshua Leung	Windows Vista 32	Intel Core2 Duo	Nvidia GeForce Go 7600
Nathan Letwory	Windows 7 Ultimate 64	AMD Turion X2 Mobile RM-74	ATI HD 4650
Nathan Letwory	Windows 7 Ultimate 64	AMD Athlon II X4 620	2x HIS ATI HD 5550 /w four monitors
Robin Allen	Windows XP 32	Intel Centrino duo	NVidia GeForce go 7600
Thomas Dinges	Windows 7 x64	Intel Core i5	Intel HD 2500
Thomas Dinges	Windows 7 x64	Intel Core i7	NVidia GeForce 540M + Intel HD 3000
Andrea Weikert	Linux 32	AMD Athlon 64 X2	Nvidia Quadro FX1500

Brecht van Lommel	Linux 64	Intel Core 2 Duo	NVidia GeForce 460 GTX
Campbell Barton	Linux 64	AMD Phenom II X6	Nvidia GeForce GTS 450
Diego Borghetti	Linux 64	Intel Core i5	Nvidia GeForce GTX 480
Diego Borghetti	Linux 64	Intel Core i7	Nvidia GeForce GTX 460M
Ken Hughes	Linux 32	Intel Core Duo	Nvidia GeForce GO 7500
Ken Hughes	Linux 64	AMD Athlon 64 X2	Nvidia GeForce 6600
Kent Mein	Linux 64	Intel Core Duo	Nvidia Quadro FX 1400
Michael Fox	Linux 32	Celeron	Nividia GeForce 6200
Raul Fernandez Hernandez	Linux 32	Pentium D 945	ATI X1550
Robin Allen	Linux 32	Intel Centrino duo	NVidia GeForce go 7600
Brecht van Lommel	OS X 10.6	Intel Core 2 Duo	NVidia GeForce 9600M GT
Dustin Martin	OSX 10.5	Dual Quad Intel	Nvidia Geforce 8800 GT
Ton Roosendaal	OSX 10.7	iMac Intel Core i7	AMD Radeon HD 6970M
Ton Roosendaal	OSX 10.8	MacBook Pro i7 "Retina"	NVidia GT 650M + Intel HD 4000
Matt Ebb	OSX 10.5	Dual Core Intel MBP	nVidia 8600M
Kent Mein	SunOS 5.8	Sun Blade 150	ATI PGX
Timothy Baldridge	SGI Irix 6.5 (mipspro)	8 x R16000	(headless)
Timothy Baldridge	SGI Irix 6.5 (mipspro)	2 x R10000	
Jeroen Bakker	Latest Ubuntu 64bit	Dell m4300 Intel Core 2 Duo 2.0Ghz	Nvidia Quadro FX360M
Sergey Sharybin	Debian Wheezy 64bit	Intel Core i7 920 2.6Ghz	Nvidia GeForce GTX 560Ti + GeForce GT 620
Sergey Sharybin	Debian Wheezy 64bit	Intel Core i5 2.4GHz	Intel Sandy Bridge + Nvidia GT 520M
Jens Verwiebe	OSX 10.6/7/8/9	Intel Xeon 6-core@ 3.33	ATI 5870/7970
Nicholas Bishop	Fedora 18 64bit	Intel Core i7 @ 2.93GHz	AMD Radeon HD 6950 (Gallium drivers, currently at OpenGL 2.1)
Nicholas Bishop	Ubuntu 12.10 64bit	Intel Core i5	ATI Mobility Radeon 5650 (Gallium drivers)
Tamito Kajiyama	Windows Vista 64bit	Intel Core2 Duo	Nvidia Quadro FX 770M
Sergej Reich	Arch Linux 64bit	Intel Core2 Quad @ 2.83GHz	Nvidia GeForce GTX 285
Sergej Reich	Arch Linux 64bit	Intel Core i3 @ 2.10GHz	Intel Sandybridge Mobile
Howard Trickey	Ubuntu 12.04 64	Intel Xeon E5-1650	NVidia Quadro 600
Howard Trickey	Windows 7 64	Intel Core i7	NVidia GeForce GTX 460
Howard Trickey	OSX 10.8.2	Intel Core Duo	NVidia GeForce 9400M

Compiling the Source

There are presently four build systems for making a binary for the different supported operating systems. Consult the Building Blender web page for more information about compiling a custom installation binary for your machine.

Python, a Scripting Language

Python Language Logo

Python is an interpreted programming language used in Blender as our main general purpose scripting language.

Python can be used to extend the functionality of Blender with regular installations available from the Blender download page. Blender comes bundled with the appropriate Python libraries, so, for regular usage a full install of Python is not required to run Python scripts.

Users wanting to write their own scripts, compile their own versions of Blender or use some less common features may still need a full Python install.

Users that want full Python functionality should refer to the Python website for installation instructions.

Installing on Linux

Download

You can obtain the latest stable version of Blender for Linux from the Blender download page or from your distribution software repository if it provides a Blender package.

Version

Blender for Linux is currently available in 32-bit and 64-bit versions. Users with a 32-bit version of Linux must download the 32-bit version of Blender. Users with a 64-bit version of Linux can choose to use either the 32-bit or 64-bit version of Blender, however you will likely notice an increase in performance when using the 64-bit version of Blender, especially on systems with large amounts of RAM.

To determine whether you have a 32-bit or 64-bit version of Linux, you can either consult your distributions' documentation or use the `uname` command with the `-m` option. `uname` will print system information and the `-m` option will print the machine hardware name.

- Open a terminal console
- Enter the command `uname -m`

If you have a 32-bit system, `uname -m` will return a value of `i686`. A 64-bit system will return a value of `x86_64`.

Distribution releases

Most major distributions such as Ubuntu, Debian, Open SUSE, Fedora and many others will provide a build of Blender in their software repository that can be accessed through that distributions package manager. If your distribution does not do this, or has not updated their repository to include the latest Blender release, you can install it yourself with the instructions below. Note that depending on your distribution, the version available in the software repository may be outdated compared to the offical release.

Installation

First check if your distribution provides the latest Blender version through its package manager. If it doesn't, download the appropriate version of Blender for Linux from the Blender download page and unpack the archive to a location of your choice.

This will create a directory named `blender-VERSION-linux-glibcVERSION-ARCH`, where `VERSION` is the Blender release version, `glibcVERSION` is the version of glibc required and `ARCH` is your computer architecture (`i686` or `x86_64`). In this directory you will find the `blender` binary.

To run Blender,

- Start your X.Org server (if it is not already running)
- Navigate to the Blender directory using a file manager and double click the Blender executable or,
- Open a terminal console, navigate to the Blender directory and execute the command `./blender`

Installing into `/opt` or `/usr/local`

You can also install Blender into `/opt` or `/usr/local` by moving the Blender directory into one of those locations. If you want to be able to run Blender from any directory you will also need to update your PATH variable. Consult your operating system documentation for the recommended method of setting your PATH.

Configuration

Alt+Mouse Conflict

Many Linux distributions default to Alt LMB 🖱 for moving windows. Since Blender uses Alt+Click it's normally easier to disable this feature or change the key to Super (Windows Key)

- Ubuntu 11.04: Settings > Window Manger Tweak > Accessibility > Change Window Key to Super
- Ubuntu 12.04 (Unity/Gnome): Command line (effective at next login): gsettings set org.gnome.desktop.wm.preferences mouse-button-modifier 'none'
- Other versions: todo

Compositing Desktop Environments

Many recent Linux distributions enable compositing when hardware support is available. This is a feature where the graphics card is used to do window drawing and accelerated desktop effects (for example: drop shadow and window transparency). Notably - Ubuntu Unity, Gnome Shell and KDE will use compositing.

While many users find this works flawlessly, some graphics cards have buggy drivers which cause drawing glitches with Blender but work correctly for regular applications which don't use OpenGL acceleration. Another downside to using hardware accelerated desktop effects is that the windows you have open share texture memory with Blender's OpenGL display and GPU rendering.

If you experience these problems they can be avoided by disabling desktop effects or by switching to a desktop environment that does not use desktop effects such as:

- Unity2D
- Gnome Fallback
- XFCE
- light weight window managers like openbox, jwm, sawfish, icewm... etc.

For details on this topic, see: Wikipedia - Compositing Window Managers

Debian based systems

Installing Blender on Debian based systems and its derivatives (Ubuntu, Mint and others), is very easy and straightforward. Most Debian distributions come with the apt-get package manager, which is powerful and solves dependencies for the packages that may need to be installed automatically.

The default install on these systems doesn't come with the libraries Blender needs to work. This is because Linux systems are planned in such a way as to only install libraries when needed, for the software that is currently installed on a system. Downloading Blender and unzipping it in the default install of most systems is not enough; The required libraries need to be installed also, copying just the Blender binary will not install the required libraries.

On this page, we explain the easiest way to install Blender for a Debian based Linux system using the default install system, which is easy and fast.

Screenshot Install

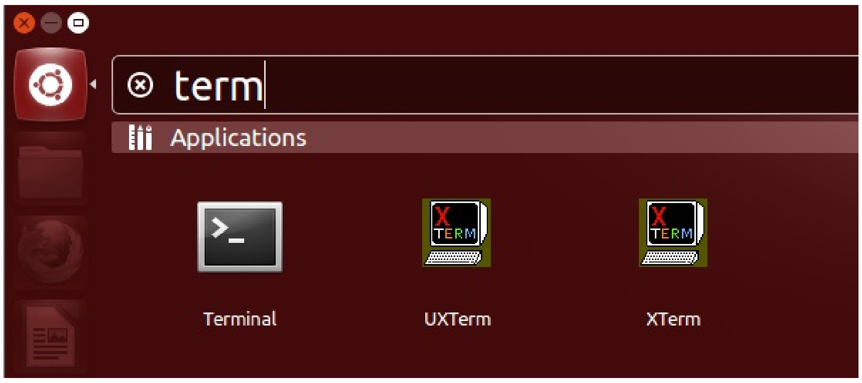
There are many different distributions based on Debian based Linux systems available to the users, and some of them use different Window managers and ways of installing software such as Blender, we can't add all of the different ways to this page. The most common way to install Blender on Debian based systems is below in text format. For the majority of users the instructions above should suffice, with little or no changes in the steps required to install Blender.

You must have administrative rights to install packages on your system

You have to be the *root/admin* user of your system, or have yourself in the *sudoers* list, or contact the system administrator to ask for administrative rights and a proper system password to install Blender and it's package dependencies, or follow the procedures on this page.

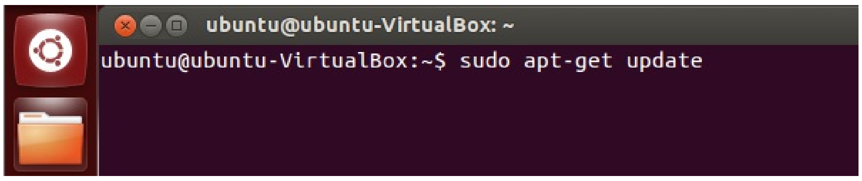
Ubuntu (step by step)

- Clicking in the Ubuntu dash, search for the terminal typing the search word **terminal**



1- Searching for the Terminal using Ubuntu dash

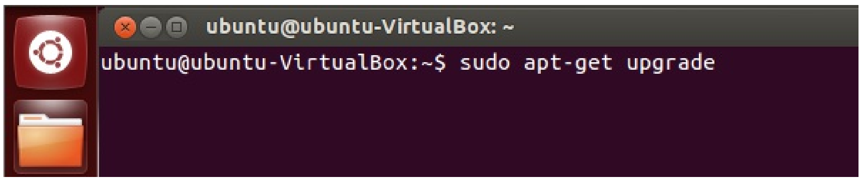
- Clicking in the *Terminal* icon, type the command **sudo apt-get update** in the prompt.
- Type the password when asked and press ↵ Enter.



```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get update
```

2- Update for the apt-get package manager list

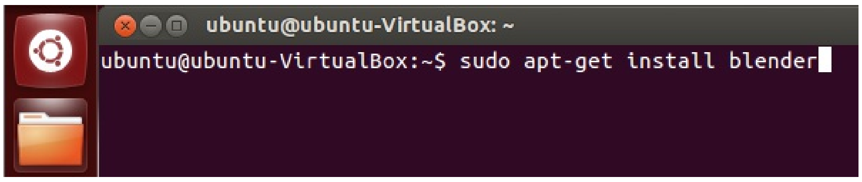
- After the update completion, type **sudo apt-get upgrade** in the prompt.
- Type the password if asked and press ↵ Enter



```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get upgrade
```

3 - System upgrade, the new list of packages for the system will be downloaded and installed

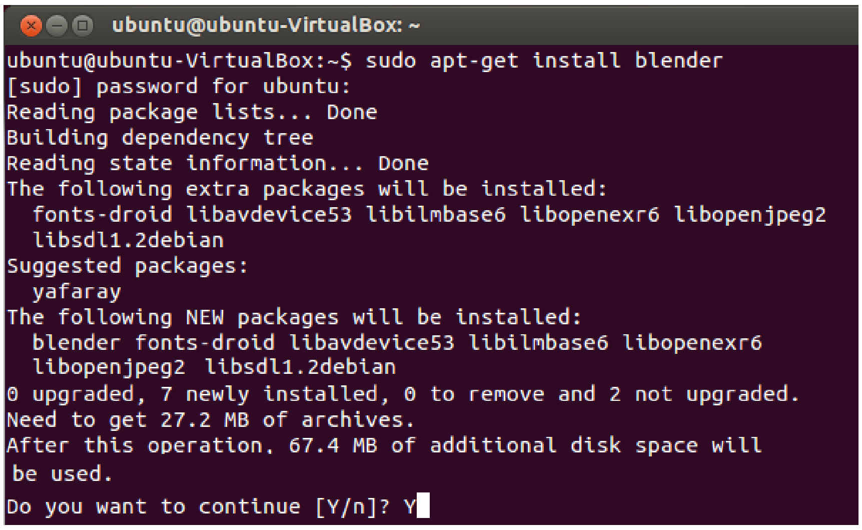
- Reboot your system, even if not asked.
- When the system is ready, open the terminal again.
- Type in the terminal **sudo apt-get install blender**,
- Type the password when asked and press ↵ Enter



```
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install blender
```

4 - Installing Blender after the updates and reboot

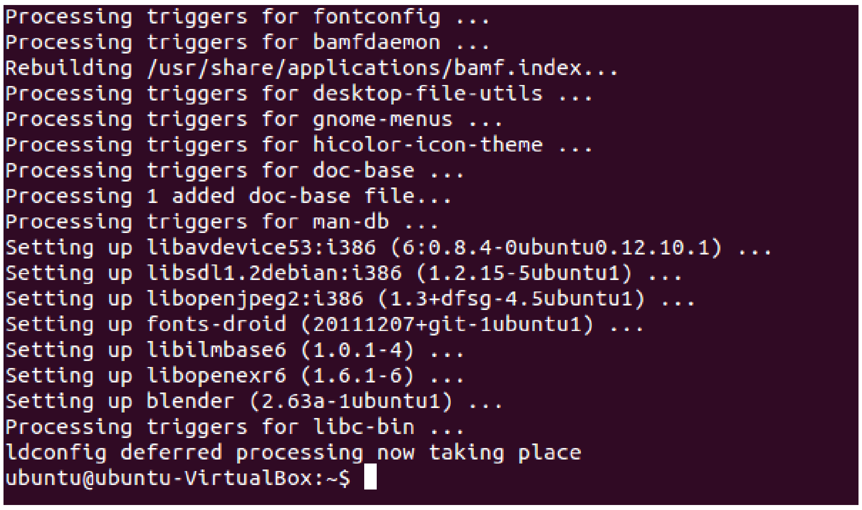
- The system will ask your permission to make changes,
- You will have to accept typing **Y** and ↵ Enter
- Blender and Blender libraries will be installed automatically.



```
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install blender
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  fonts-droid libavdevice53 libilmbase6 libopenexr6 libopenjpeg2
  libsdl1.2debian
Suggested packages:
  yafaray
The following NEW packages will be installed:
  blender fonts-droid libavdevice53 libilmbase6 libopenexr6
  libopenjpeg2 libsdl1.2debian
0 upgraded, 7 newly installed, 0 to remove and 2 not upgraded.
Need to get 27.2 MB of archives.
After this operation, 67.4 MB of additional disk space will
 be used.
Do you want to continue [Y/n]? Y
```

5 - System asking your permission to make changes

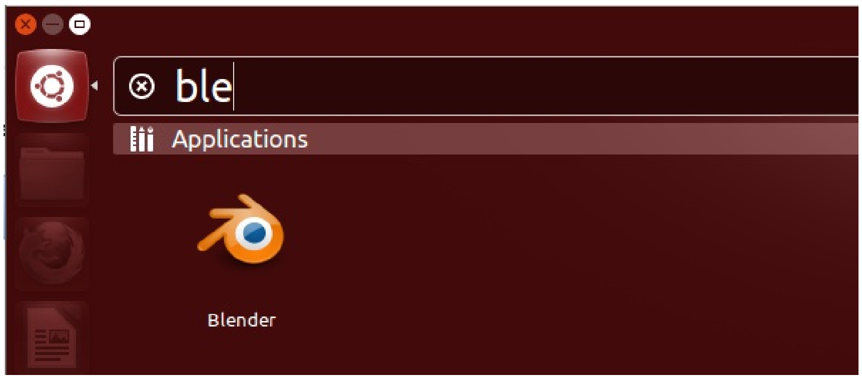
- The apt-get package manager will do everything that is needed to install Blender and its dependencies.



```
Processing triggers for fontconfig ...
Processing triggers for bamfdaemon ...
Rebuilding /usr/share/applications/bamf.index...
Processing triggers for desktop-file-utils ...
Processing triggers for gnome-menus ...
Processing triggers for hicolor-icon-theme ...
Processing triggers for doc-base ...
Processing 1 added doc-base file...
Processing triggers for man-db ...
Setting up libavdevice53:i386 (6:0.8.4-0ubuntu0.12.10.1) ...
Setting up libsdl1.2debian:i386 (1.2.15-5ubuntu1) ...
Setting up libopenjpeg2:i386 (1.3+dfsg-4.5ubuntu1) ...
Setting up fonts-droid (20111207+git-1ubuntu1) ...
Setting up libilmbase6 (1.0.1-4) ...
Setting up libopenexr6 (1.6.1-6) ...
Setting up blender (2.63a-1ubuntu1) ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
ubuntu@ubuntu-VirtualBox:~$
```

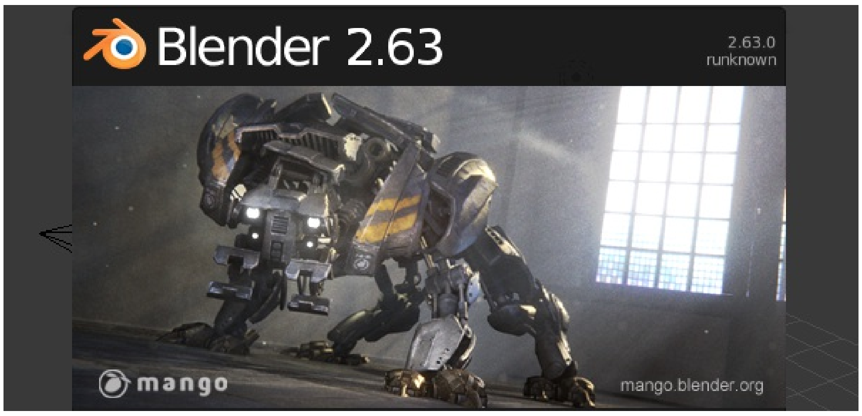
6 - Blender and its dependencies Installed

- Now, you can search for Blender using the Ubuntu dash and typing the search word **blender**.
- Click in Blender,
- If Blender is working, now you can download the newest pre-compiled Blender version.



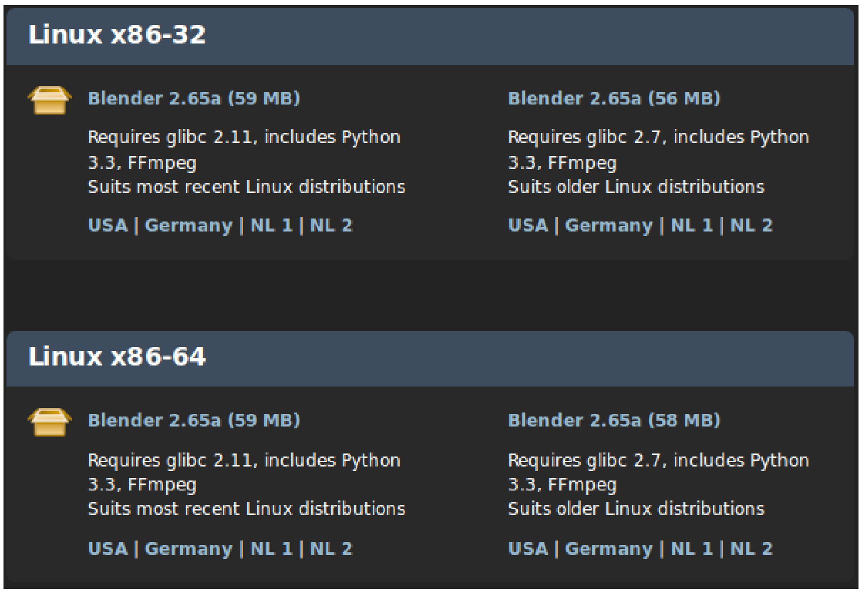
7 - Searching Blender using the Ubuntu Dash

- Blender 2.63 is the pre-compiled version used in Ubuntu 12.10.
- Now, we have to use the **uname - a** command in the terminal and see if the system is a 32 or 64 bit.
- If the command prints **i686**, your system is using a 32 bit Linux version.
- If the command prints **x86_64**, your system is using a 64 bit Linux version.



8 - Blender 2.63 is the pre-compiled version bundled with Ubuntu 12.10

- Opening blender.org's website, you will see the pre-compiled versions for your Linux system.
- Download the newest Blender version, clicking on the *suits most recent linux versions* column
- Choose the appropriate version for your system (32 or 64 bits)



Linux x86-32

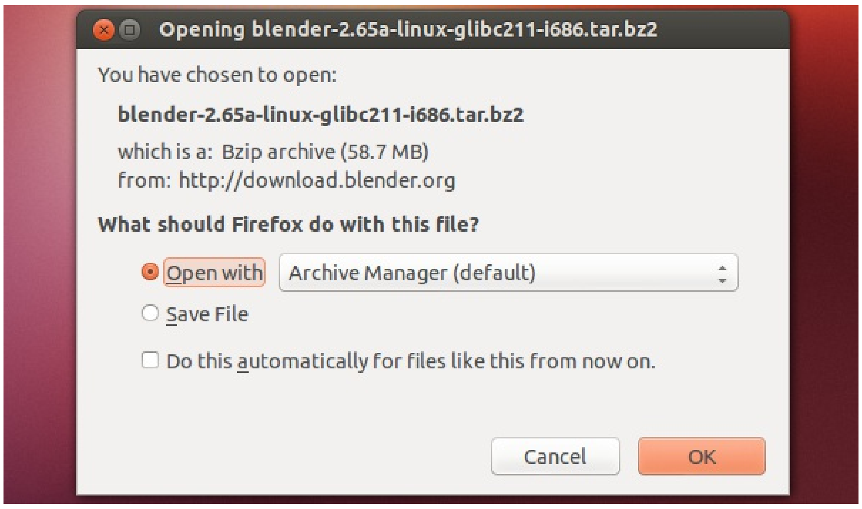
Blender 2.65a (59 MB) Requires glibc 2.11, includes Python 3.3, FFmpeg Suits most recent Linux distributions **USA | Germany | NL 1 | NL 2**	**Blender 2.65a (56 MB)** Requires glibc 2.7, includes Python 3.3, FFmpeg Suits older Linux distributions **USA | Germany | NL 1 | NL 2**

Linux x86-64

Blender 2.65a (59 MB) Requires glibc 2.11, includes Python 3.3, FFmpeg Suits most recent Linux distributions **USA | Germany | NL 1 | NL 2**	**Blender 2.65a (58 MB)** Requires glibc 2.7, includes Python 3.3, FFmpeg Suits older Linux distributions **USA | Germany | NL 1 | NL 2**

9 - Pre-compiled versions of Blender for Linux on blender website

- Your browser will ask you what to do with the zipped file.
- Choose **Open with - Archive manager (default)**


Opening blender-2.65a-linux-glibc211-i686.tar.bz2

You have chosen to open:

blender-2.65a-linux-glibc211-i686.tar.bz2

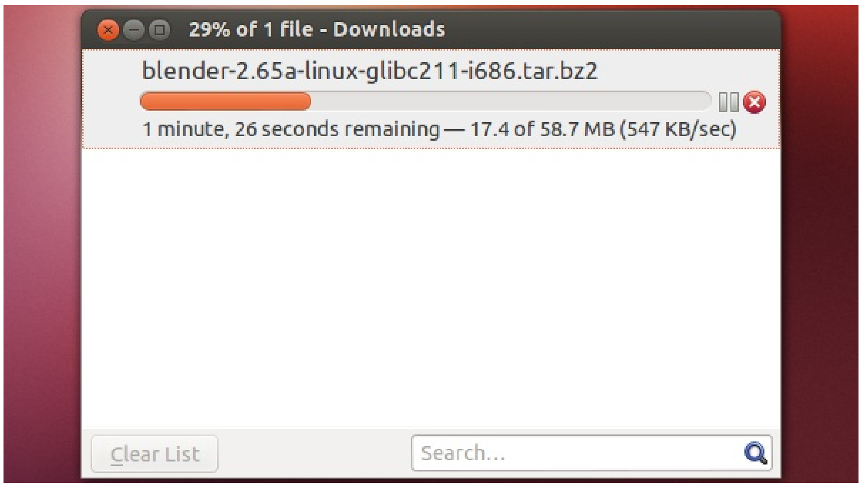
which is a: Bzip archive (58.7 MB)
from: http://download.blender.org

What should Firefox do with this file?

[✓ Open with] Archive Manager (default)
Save File
Do this automatically for files like this from now on.

10 - Browser asking you to choose an action for the zipped Blender archive

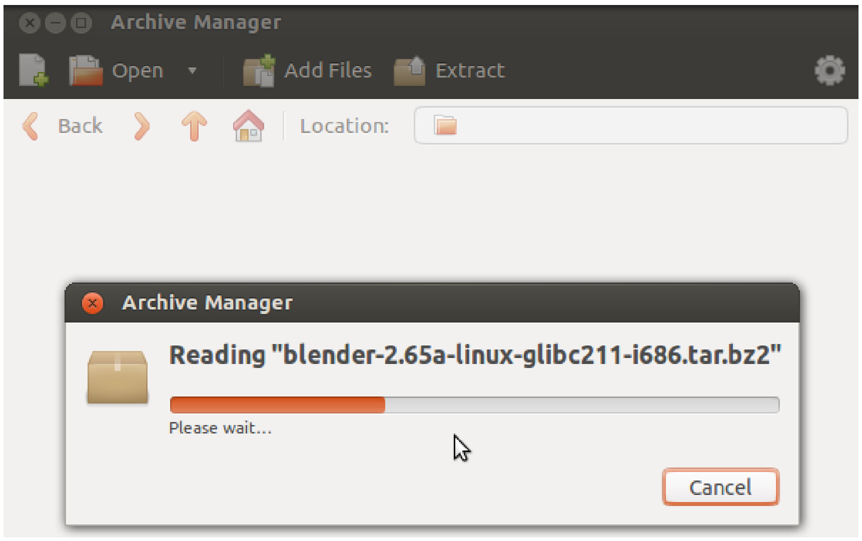
- Wait for Blender to download.


29% of 1 file - Downloads

blender-2.65a-linux-glibc211-i686.tar.bz2
1 minute, 26 seconds remaining — 17.4 of 58.7 MB (547 KB/sec)

11 - Browser downloading pre-compiled version of Blender from blender.org website

- The zipped Blender archive will be read by your archive manager automatically.



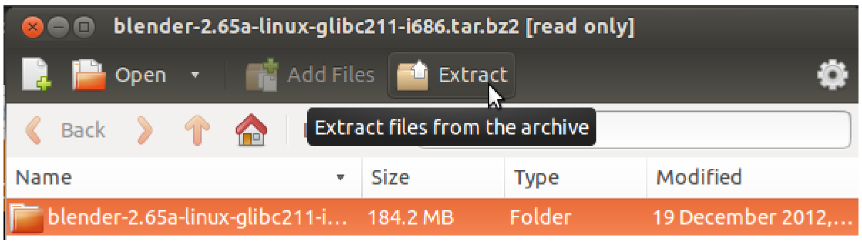
Archive Manager

Reading "blender-2.65a-linux-glibc211-i686.tar.bz2"

Please wait...

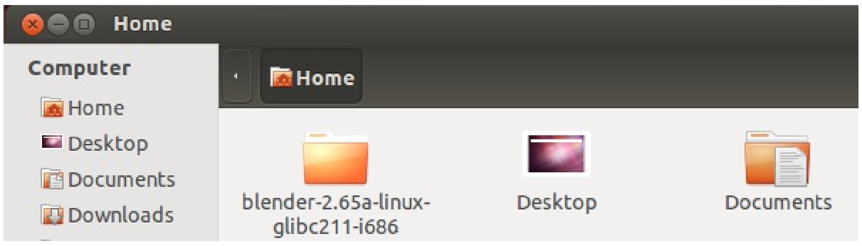
12 - Ubuntu archive Manager opening zipped Blender

- After the proccess completion, you will be presented with a folder.
- Click to select and click in the extract button.
- The default is to extract to the user home folder.


blender-2.65a-linux-glibc211-i686.tar.bz2 [read only]
Open
Add Files
Extract
Back
Extract files from the archive
Name
Size
Type
Modified
blender-2.65a-linux-glibc211-i...
184.2 MB
Folder
19 December 2012,...

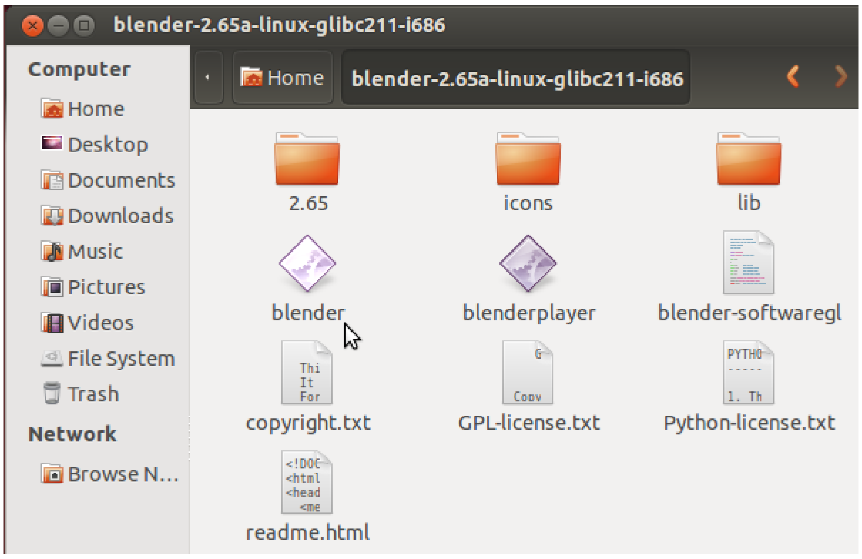
13 - Extract button, this will extract Blender to a folder

- After the extraction, you can close the Ubuntu archive manager if it's not closed automatically.
- Go to your Home folder and you will see a new folder extracted with **blender...*** name.
- Click on this folder to open.



14 - Blender folder extracted shown at the user home folder

- Now you can see the extracted contents of the Blender package in the folder.
- Click on Blender and you will have the newest Blender version working.
- At the time we made this page, the newest Blender version was 2.65a.



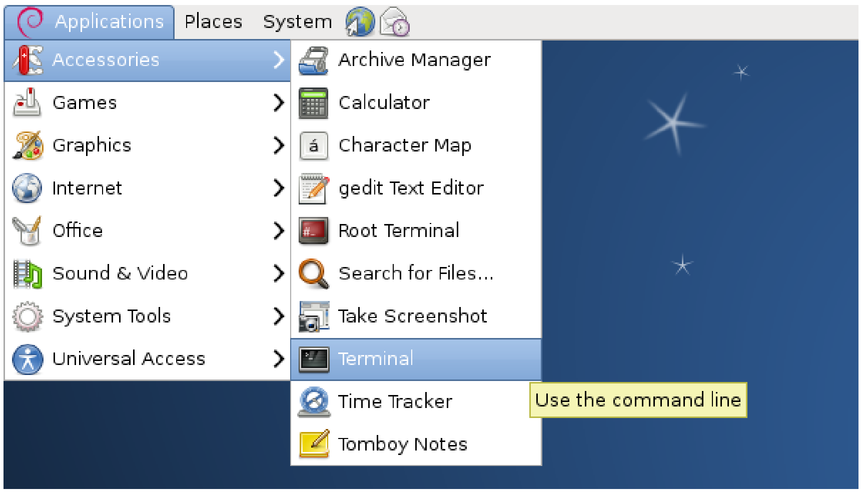
15 - Newest Blender Binary extracted and ready for execution

Debian (step by step)

You must have administrative rights to install packages on your system

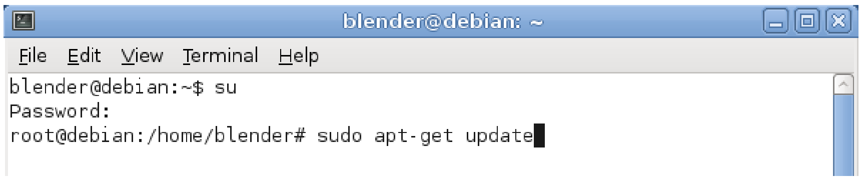
You have to be the *root/admin* user of your system, or have yourself in the *sudoers* list, or contact the system administrator to ask for administrative rights and a proper system password to install Blender and it's package dependencies, or follow the procedures on this page.

- Clicking in the applications menu, search for the **Terminal** in the **Acessories** entry.



1 - Terminal in the **Acessories** entry - Debian default install.

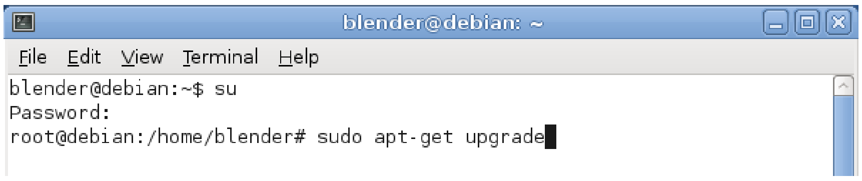
- Clicking in the *Terminal* icon, type the command **sudo apt-get update** in the prompt.
- Type the password when asked and press ↵ Enter.



```
blender@debian:~$ su
Password:
root@debian:/home/blender# sudo apt-get update
```

2 - Update for the apt-get package manager list

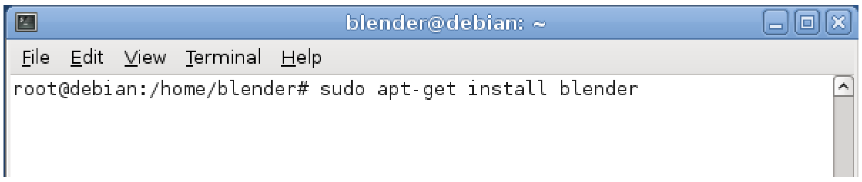
- After the update completion, type **sudo apt-get upgrade** in the prompt.
- Type the password if asked and press ↵ Enter



```
blender@debian:~$ su
Password:
root@debian:/home/blender# sudo apt-get upgrade
```

3 - System upgrade, the new list of packages for the system will be downloaded and installed

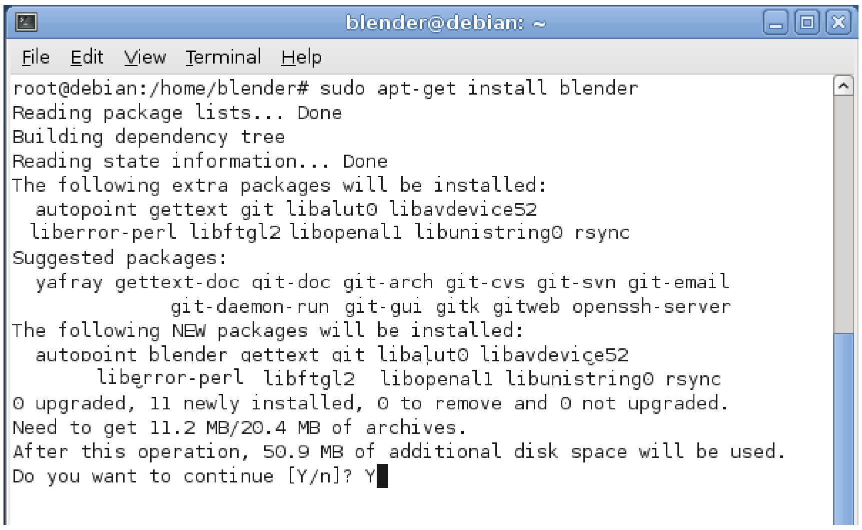
- Reboot your system, even if not asked.
- When the system is ready, open the terminal again.
- Type in the terminal **sudo apt-get install blender**,
- Type the password when asked and press ↵ Enter



```
root@debian:/home/blender# sudo apt-get install blender
```

4 - Installing Blender after the updates and reboot.

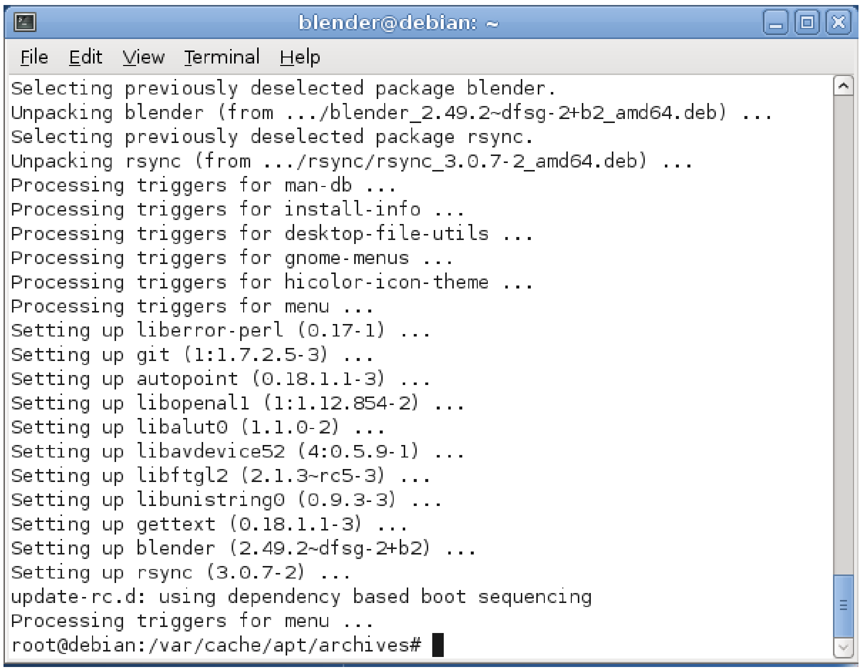
- The system will ask your permission to make changes,
- You will have to accept typing **Y** and ↵ Enter
- Depending on your install method and package repository, you system mays ask you an install CD/DVD.
- Insert your CD/DVD disc and press ↵ Enter
- Blender and Blender libraries will be installed automatically.



```
root@debian:/home/blender# sudo apt-get install blender
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  autopoint gettext git libalut0 libavdevice52
 liberror-perl libftgl2 libopenal1 libunistring0 rsync
Suggested packages:
  yafray gettext-doc git-doc git-arch git-cvs git-svn git-email
              git-daemon-run git-gui gitk gitweb openssh-server
The following NEW packages will be installed:
  autopoint blender gettext git libalut0 libavdevice52
       liberror-perl  libftgl2  libopenal1 libunistring0 rsync
0 upgraded, 11 newly installed, 0 to remove and 0 not upgraded.
Need to get 11.2 MB/20.4 MB of archives.
After this operation, 50.9 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

5 - System asking your permission to make changes

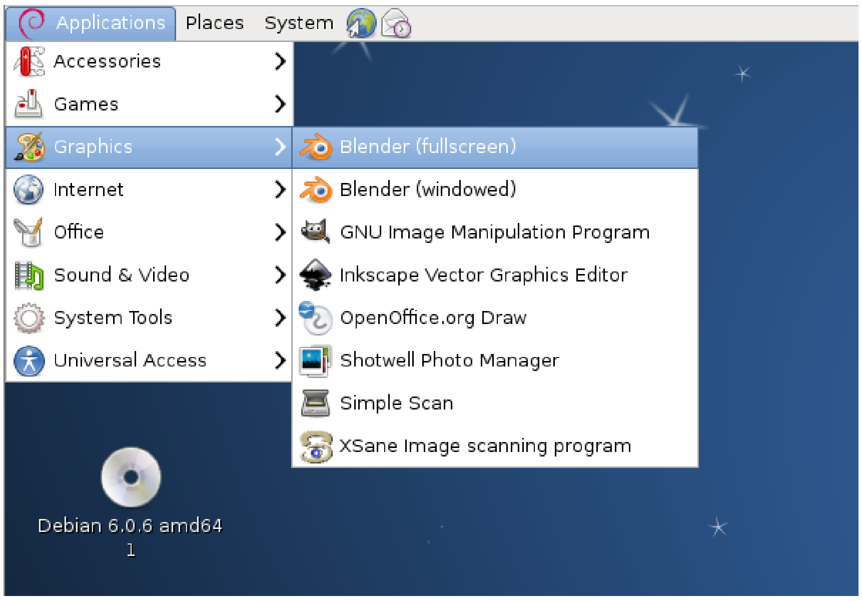
- The apt-get package manager will do everything that is needed to install Blender and its dependencies.



```
Selecting previously deselected package blender.
Unpacking blender (from .../blender_2.49.2~dfsg-2+b2_amd64.deb) ...
Selecting previously deselected package rsync.
Unpacking rsync (from .../rsync/rsync_3.0.7-2_amd64.deb) ...
Processing triggers for man-db ...
Processing triggers for install-info ...
Processing triggers for desktop-file-utils ...
Processing triggers for gnome-menus ...
Processing triggers for hicolor-icon-theme ...
Processing triggers for menu ...
Setting up liberror-perl (0.17-1) ...
Setting up git (1:1.7.2.5-3) ...
Setting up autopoint (0.18.1.1-3) ...
Setting up libopenal1 (1:1.12.854-2) ...
Setting up libalut0 (1.1.0-2) ...
Setting up libavdevice52 (4:0.5.9-1) ...
Setting up libftgl2 (2.1.3~rc5-3) ...
Setting up libunistring0 (0.9.3-3) ...
Setting up gettext (0.18.1.1-3) ...
Setting up blender (2.49.2~dfsg-2+b2) ...
Setting up rsync (3.0.7-2) ...
update-rc.d: using dependency based boot sequencing
Processing triggers for menu ...
root@debian:/var/cache/apt/archives#
```

6 - Blender and its dependencies Installed

- Now, you can search for Blender in the applications menu, in the **Graphics** entry .
- Click on Blender,
- If Blender is working, now you can download the newest pre-compiled Blender version.

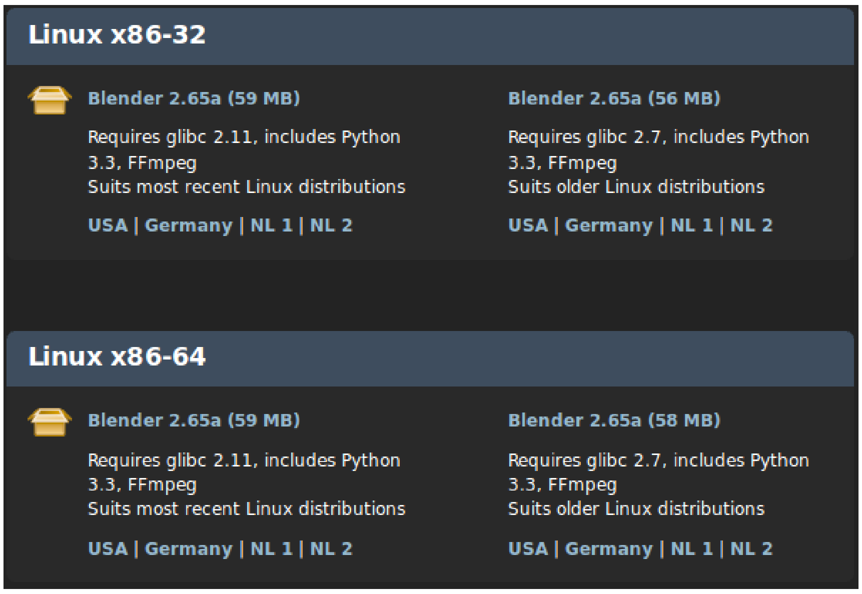


7 - Blender in the graphics entry

- Blender 2.49b is the pre-compiled version used in Debian 6.06.
- Now, we have to use the **uname - a** command in the terminal and see if the system is a 32 or 64 bit.
- If the command prints **i686**, your system is using a 32 bit Linux version.
- If the command prints **x86_64**, your system is using a 64 bit Linux version.

8 - Blender 2.49b is the pre-compiled version used in Debian 6.06.

- Opening blender.org's website, you will see the pre-compiled versions for your Linux system.
- Download the newest Blender version, clicking on the suits most recent linux versions column
- Choose the appropriate version for your system (32 or 64 bits)


Linux x86-32

Blender 2.65a (59 MB)
Requires glibc 2.11, includes Python 3.3, FFmpeg
Suits most recent Linux distributions
USA | Germany | NL 1 | NL 2

Blender 2.65a (56 MB)
Requires glibc 2.7, includes Python 3.3, FFmpeg
Suits older Linux distributions
USA | Germany | NL 1 | NL 2

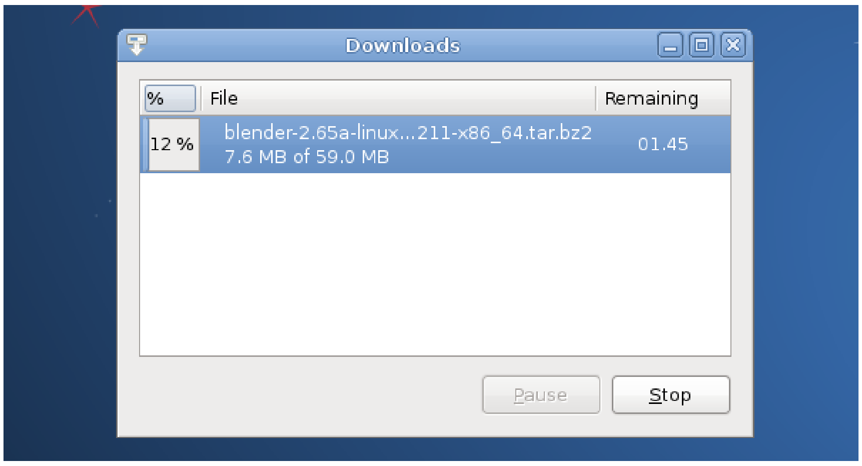
Linux x86-64

Blender 2.65a (59 MB)
Requires glibc 2.11, includes Python 3.3, FFmpeg
Suits most recent Linux distributions
USA | Germany | NL 1 | NL 2

Blender 2.65a (58 MB)
Requires glibc 2.7, includes Python 3.3, FFmpeg
Suits older Linux distributions
USA | Germany | NL 1 | NL 2

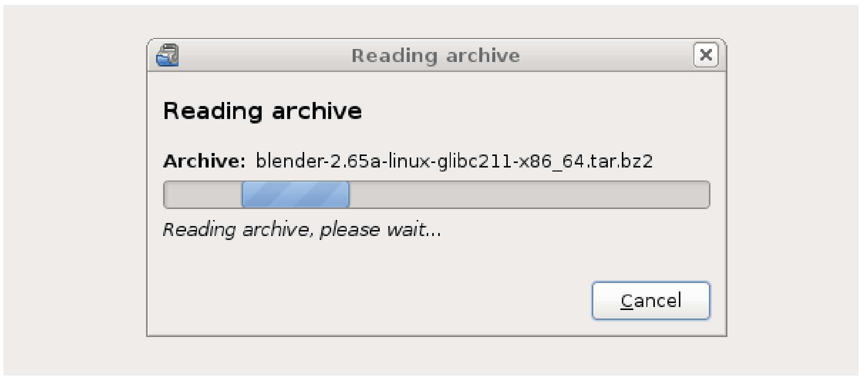
9 - Pre-compiled versions of Blender for Linux on blender website

- Your browser will ask you what to do with the zipped file.
- Choose Open with - Archive manager (default) or...
- If no actions are asked, click on the Blender file when the download is complete.
- Wait for Blender to download.



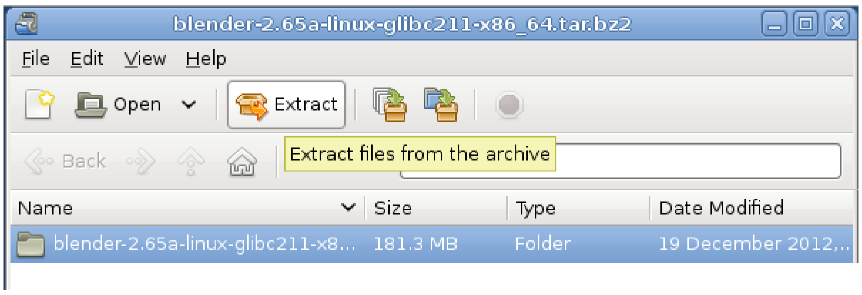
10 - Browser asking you to choose an action for the zipped Blender archive

- The zipped Blender archive will be read by your archive manager automatically


Reading archive
Reading archive
Archive: blender-2.65a-linux-glibc211-x86_64.tar.bz2
Reading archive, please wait...
Cancel

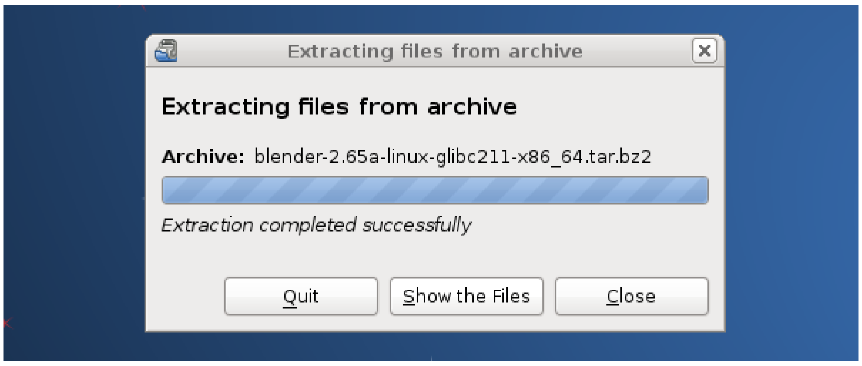
11 - The zipped Blender archive will be read by your archive manager automatically

- After the proccess completion, you will be presented with a folder.
- Click to select and click in the extract button.
- The default is to extract to the user **Download** folder, located at the user's home folder.


blender-2.65a-linux-glibc211-x86_64.tar.bz2
File
Edit
View
Help
Open
Extract
Back
Extract files from the archive
Name
Size
Type
Date Modified
blender-2.65a-linux-glibc211-x8...
181.3 MB
Folder
19 December 2012,...

12 - Extract button, this will extract Blender to a folder

- Debian archive manager will extract Blender
- When the extraction proccess is complete with success, click in the **Quit** button



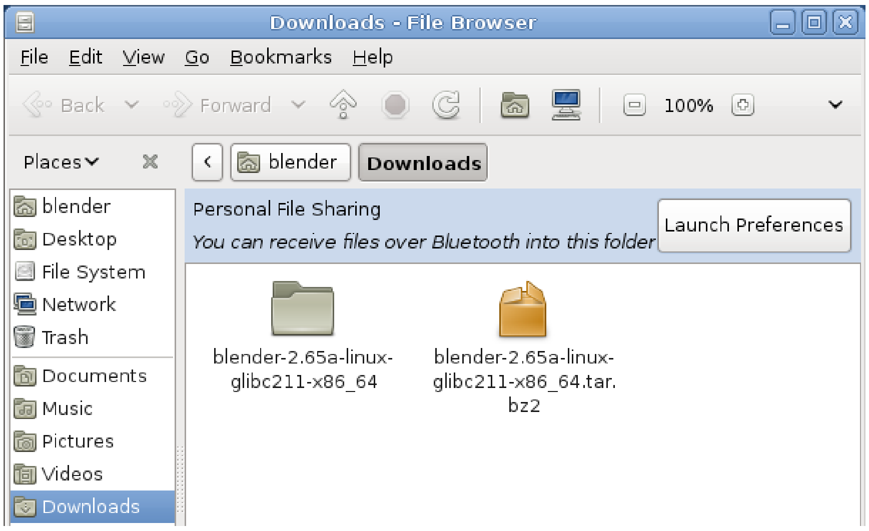
Extracting files from archive

Archive: blender-2.65a-linux-glibc211-x86_64.tar.bz2

Extraction completed successfully

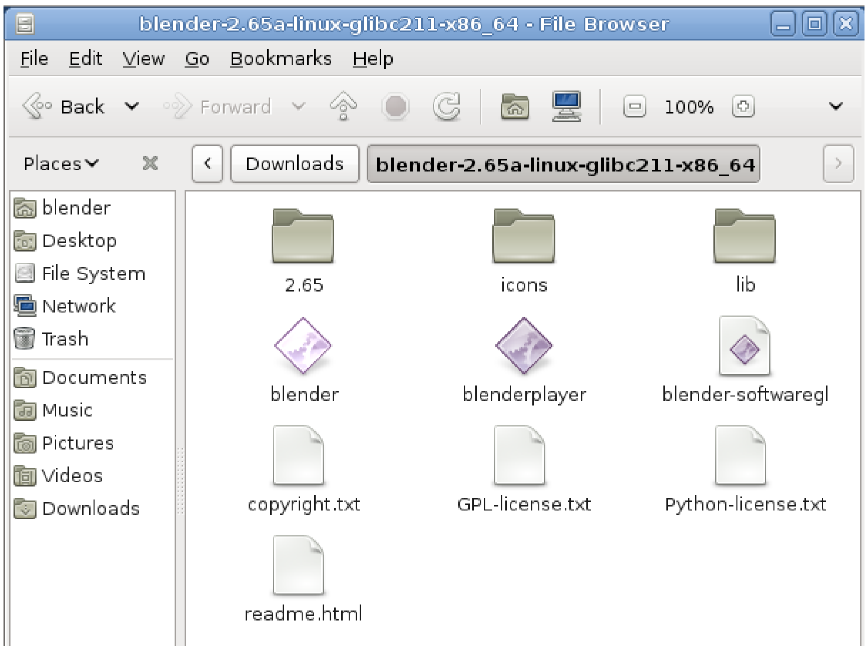
13 - Debian archive Manager extracting Blender, finished

- After the extraction, you can close the Debian archive manager if it's not closed automatically.
- Go to your Home folder and you will see a new folder extracted with **blender...*** name.
- Click on this folder to open.



14 - Blender folder extracted shown at the user **Downloads** folder

- Now you can see the extracted contents of the Blender package in the folder.
- Click on Blender and you will have the newest Blender version working.
- At the time we made this page, the newest Blender version was 2.65a.



15 - Newest Blender Binary extracted and ready for execution

General Instructions (text)

You must have administrative rights to install packages on your system

You have to be the *root/admin* user of your system, or have yourself in the *sudoers* list, or contact the system administrator to ask for administrative rights and a proper system password to install Blender and it's package dependencies, or follow the procedures on this page.

- **Those instructions were tested for Blender 2.65 using Debian 6.0, Ubuntu 12.04 and 12.10**.
- In some Debian based systems, you don't have the sudo command enabled by default, so you will have to type **su**, and type the system password to be logged as *root* first and type **apt-get update** after, then you can continue by entering the following the commands:

- With the default install, open your terminal by clicking the terminal icon for your Linux terminal or console of your system.
- Type in the terminal:

```
sudo apt-get update
```

- The system will require the *root/admin* password. Type your password and press ↵ Enter and wait for the system to update the file list of the apt package manager.
- After the update, type in the terminal:

```
sudo apt-get upgrade
```

- Press ↵ Enter
- Depending on the amount of time the update took, your system may require your password again. Type your password, press ↵ Enter and wait for the apt-get package manager, to download and update all installed packages on your system (system update).
- Your system may ask to reboot, even if the system doesn't ask you to reboot, it's better to do so, because the most recent kernel and new libraries will be used after the reboot.
- After the reboot, again open your Linux Terminal or console.
- Type in the terminal:

```
sudo apt-get install blender
```

- Press ↵ Enter
- The apt-get package manager will then install the current pre-compiled version of Blender for your Debian based system. It will automatically install all the required libraries and/or dependencies as well.

- Now you will probably have a working Blender version installed and its dependencies. You can search for the newly installed Blender version in your system menus, or by using your system's search feature, or by using the command line. You should test to see if it will run correctly. If Blender is running correctly (even if it's an outdated version), then you're ready to download the latest Blender version.

- Blender is provided in 2 different formats, a 32bit version of Blender and 64bit version of Blender. Prior to downloading a particular version of Blender, you need to know which version of Blender you need. To find out type the following command in your Linux terminal:

```
uname -a
```

- If your system prints a message on the console screen showing **i686**, you have a 32 bit system, if your system prints a message on the console screen showing **x86_64**, then you have a 64 bit system. Now, you can download an appropriate Blender version for your system. If your system is 32 bit you must download the 32 bit version of Blender. If your system is 64 bit, then you can download the 64 bit version of Blender. Also note that 64 bit platforms can also run 32 bit versions of Blender but this will mean you will not be able to access any memory in your system above 4 gigabytes, and 32 bit version of Blender will perform more slowly on 64 bit platforms.

- Go to the blender.org download website and download the correct Blender Linux version for your system. To Download Blender there is a column on the website marked with *Suits most recent Linux distributions* on Blender.org's website.

- The Blender pre-compiled packages from blender.org for Debian/Linux based systems come packaged in a zip file. You can choose to download and unzip to a folder after the download, or open it with your Archive Manager (default) when asked by your internet browser.

- After the download, unzip the file that is shown in your archive manager into another folder. After successfully unzipping the file, open the location where you have unzipped Blender using your file manager.

- Locate and click twice on blender or blender.bin and you should see latest working version of Blender start to execute and display the Blender Splash Screen!

Hints

- Installing newest Blender version into `/opt` or `/usr/local`

You can also install Blender into `/opt` or `/usr/local` by moving the Blender directory into one of those locations. If you want to be able to run the newest Blender from any directory you will also need to update your PATH variable. Consult your operating system documentation for the recommended method of setting your PATH.

- You can use the contents of the Blender archive and copy over you old Blender install.

You can use the extracted contents of the downloaded Blender archive (newest), and copy the contents over your distribution install, using your *admin/root* credentials, for example in the `/usr/bin/` folder, but be aware that you will have to cleanup the old blender folders everytime you update.

Drivers for 3D Graphic Cards

To run 3D software packages such as Blender, your system will need several specialized software libraries which interpret 3D drawing commands from Blender into drawing commands for your computer screen and graphics card.

Blender uses OpenGL which is free graphics language library that works on multiple platforms. The OpenGL drivers can be implemented in 2 different ways in Linux:

- Via Software - You have software such as MesaGL which is a software library that uses your CPU to interpret OpenGL commands and convert those commands into pixels that get displayed on your screen. Those commands will use your CPU to processes the OpenGL 3D drawing commands, which will then be drawn upon your screen. Interpreting the OpenGL 3D Drawing commands with your CPU is much slower and less efficient and so will result in slower 3D drawing display performance in software such as Blender. This results in for example your 3D Viewport not displaying models as quickly or smoothly updating when doing modeling for very vertex heavy models.
- Via Hardware - When OpenGL drawing commands are processed in hardware, the drawing commands are sent directly to your 3D graphics card hardware. The CPU is bypassed for the most part and this results in a much greater performance level when displaying 3D data such as mesh models in Blender's 3D Viewport. 3D display command processing is also called 3D Graphics Hardware Acceleration.

Most modern Linux distributions, including Debian, come with MesaGL or other OpenGL libraries bundled so you can run 3D package software such as Blender, without having specialized hardware accelerated graphics card to calculate screen drawing commands. Most modern computers nowadays come with specialized hardware which you can use to speed up the display or your 3D graphics data.

For graphic card accelerators, you have two choices to enable their full potential, use open sourced drivers or proprietary ones.

Open Sourced drivers are detected automatically for Linux based systems if your graphics card is supported by the Linux community. Some graphics card manufactures make available graphic card api's and source code, allowing the Linux community to write graphics card drivers for those cards, allowing Linux to communicate reliably and efficiently with those graphics cards. This mean that those cards perform very well on Linux.

Proprietary drivers needs the user to install third party software, which aren't Open Source (meaning no source code is released). These drivers are released by the manufacture in binary only format and they are in control of what features the driver supports for a particular graphic card. These binary only software drivers can't be *read* by the Linux community as a whole and problems/instabilities can't be fixed by Linux programmers/engineers. So, there are advantages and disadvantages when using proprietary drivers. The advantage is that you will be able to use your graphic card to speed up your work flow, the disadvantages are related to software updates, fixes, and general support.

When using Debian based systems, some distributions such as Ubuntu facilitate the proprietary driver installation using systems such as Ubuntu *proprietary drivers* (available to the majority of *buntu variants), while others will need the user to compile the manufacturers card drivers to be able to use the hardware graphic accelerated features of a particular graphic card.

Consult your Linux documentation and your card manufacturer documentation to know how to install proprietary drivers. If you find problems when using proprietary drivers, contact your card manufacturer, they are the only ones enabled to make fixes and give users support for their closed source drivers and cards.

- Proprietary drivers are an exception rather than the rule in the Linux world.

SoftwareGL Mode

Hardware or Software OpenGL Mode

There are 2 different ways of starting Blender. The first way is in Hardware Accelerated OpenGL mode, in this mode if your graphics card has Hardware support for OpenGL drawing commands Blender will use it. Blender will perform much more quickly when it is run in Hardware Accelerated OpenGL Mode. To start Blender in Hardware Accelerated OpenGL Mode type the following command at the terminal:

```
./blender
```

Some graphics cards either don't work at all or don't display information in Blender correctly when run this way. If this happens for you then you can run Blender in Software OpenGL Mode. To do this start Blender from the terminal by typing:

```
./blender-softwaregl
```

When started in this way Blender will use your CPU to process OpenGL drawing commands rather than using the dedicated hardware on your graphics card. This will result in Blender performing more slowly when doing 3D graphical tasks but it often will enable Blender to display correctly when it would not otherwise.

Cycles Rendering

Cycles is the new rendering engine in development for Blender, at first, it was a project for realtime visualization, but now its being developed as a substitute to the Blender Internal renderer.

Linux based systems and Blender fully support the use of multiple cpu's/gpu to spread render tasks in Cycles. Appropriate drivers are all that is required for the particular hardware to shared between multiple devices.

Cycles can use system CPUs (including multithreaded CPUs) or use an array of processors present in some graphic cards (GPUS) or specific processing cards to improve rendering speed, so you can choose, depending on your system and drivers, to render your images using the CPU processors or those present in your GPUS or processing cards, but you will need specific cards which are manufactured with capable processors and use appropriate drivers. Currently CUDA based hardware acceleration (as used by NVIDIA graphics cards) has the most support in Blender. Hopefully OpenCL based hardware acceleration support will develop from its current state of instability.

Blender will automatically detect your array of processor devices for Cycles if you have a capable graphics card or processing card and appropriate drivers.

As a General rule, if you have installed appropriate drivers and your graphics card or processing card is capable of using an array of processors to speed rendering with Cycles, you will be able to enable them by opening Blender User preferences Window with shortcut CtrlAltU. In the *System* tab, you will find the *Compute Device* buttons. These buttons are enabled automatically if you have a graphics card or a processing card and appropriate drivers.

For now, the only graphic card and processor card brand that works well with Cycles rendering is Nvidia, and the only available API (Aplication Programmable Interface) available to Blender is Cuda. If the *Cuda* button (for Nvidia Graphic cards) is enabled, then you have a capable graphic card or array of processors card and appropriate drivers from Nvidia installed in your Linux based system.

- For now, there are no free drivers available to Linux customers to use with cards manufactured with arrays of processors.
- CUDA is Nvidia proprietary, and there are no free drivers available to customers for now, so, the only way to enable CUDA is to have a Nvidia card and proprietary drivers installed on your Linux based system.
- There are other GPU card manufacturers with processor arrays that are capable for Cycles rendering, but their drivers and/or API are outdated and *buggy* for Linux based systems, including Debian.

Solving problems

Most Linux distributions when installed properly, works flawlessly with Blender. Minor problems are found depending on the distribution and its configuration. If Blender doesn't work, you may have to see your specific Linux distribution documentation and/or ask your system administrator to help you.

The most common cause of problems are shown here with possible solutions:

Shortcut Conflicts

Many Linux distributions default to Alt LMB for moving windows. Since Blender uses Alt+Click it's normally easier to disable this feature or change the key to Super key (In most keyboards, printed as *Windows* Key)

- Ubuntu 11.04: Settings > Window Manger Tweak > Accessibility > Change Window Key to Super
- Ubuntu 12.04 (Unity/Gnome): command line (effective at next login): gsettings set org.gnome.desktop.wm.preferences mouse-button-modifier 'none'
- Debian 7 "Wheezy" (Gnome): you need to have dconf-editor installed and go to org → gnome → desktop → wm → preferences and change "mouse-button-modifier" to 'none'. The above command line doesn't work in all situations.

Desktop Effects

Sometimes, effects and composition such as compiz , metacity, clutter, depending on your system, are resource hungry and heavy to use in conjunction with 3D package software.

Some Debian based distributions like Ubuntu, enables desktop effects *out of the box*, while others, uses a lightweight window manager which uses less resources from your system and graphic card.

If you're experiencing problems, flickering during window transitions, window fades shown at a *frame by frame* rate and others, you may have to disable your desktop effects prior to use 3D software or use another window manager without desktop effects enabled.

Desktop effects and 3D Packages

As a general rule, the best usage scenario for Blender (as with any other production 3D package software), is to have all possible system resources free, available and ready for use, and it means you will have the best possible experience using your system without desktop effects.

- Ubuntu:

There is no easy way *out of the box* to disable the desktop effects that comes with Ubuntu default install, because there is no shortcut, icon or preferences tab available to disable desktop effects for the users.

The easiest way to improve 3D package software experiences when using Ubuntu with Unity (default), is to follow the instructions below.

- Find the Terminal or console in your system and type:

```
sudo apt-get install compizconfig-settings-manager
```

- Once installed, go to Ubuntu Unity Plugin → Experimental (Tab)
- From there you can set Launch Animation, Urgent Animation and Dash Blur to 'None'.
- Set the Hide Animation to Slide only.
- If you want, you can change the panel and dash transparency to be full opaque (recommended).

External link (askubuntu.com) :

Disabling Ubuntu Desktop animations

You can also use another **buntu* distribution (Like Xubuntu or Lubuntu), that uses another lightweight window manager, like the Xubuntu variant or install another Window manager in conjunction with your default Ubuntu install.

Consult the Ubuntu documentation, or ask your system administrator on how to install another Window manager with no desktop effects to improve your 3D package experience.

- For other Debian based systems:

In general, if you don't have a composite window manager installed using desktop effects in conjunction with your window manager, you don't have to worry about it.

If you have the Compiz or Metacity, Clutter composite manager installed, consult the documentation of your composite manager to know how to disable desktop effects through shortcuts. This will improve your 3D package software experience.

Consult your system documentation or internet resources to know how to disable desktop effects for your Debian based system and make all of the available resources ready for your production 3D package.

Intel Graphic Cards

Intel is currently the largest supplier of Integrated 3D Graphics chips in the world that go inside Laptop machines and Server boards.

Unfortunately they are not very good performance graphics hardware. Not only are they often very slow, they also often do not properly implement certain 3D Graphics OpenGL commands. That can result in screen items not being displayed correctly when Blender is being used.

The only real solution when you can't use graphic accelerator expansion cards is to always keep your Intel graphics card drivers up to date and hope that the updated driver fixes any issues you may have.

Compiling Blender

If you want to build Blender from source code so you can get the latest greatest features of Blender, you can follow the official instruction. Building Blender from source is not difficult when compared with other software building proccess, but it takes some preparation and configuration to get it right. If you take your time and read all the instructions, you should be able to do it.

- Developer instructions for building blender binary from sources

If you still need help with Blender coding and compiling proccess and have tried an internet search first but with no answer, then you can always goto the irc server irc.freenode.net #blendercoders channel and report the problem you are having. The coders are busy so they can take a while to help but they will do in general. If you don't have an irc client on your machine you can click the following link and that will connect you to irc through your web browser:

- irc.freenode.net #blendercoders channel

Useful links

If you want to get versions of Blender which are more up to date as they are built from a current snapshot of the Blender SVN trunk periodically, you have a couple of websites you can use:

The graphicall.org website is a Blender users site where many different snap shots of Blender Source code are compiled by users and made available for download. This website has many builds of Blender with very experimental features enabled.

- www.graphicall.org

The builder.blender.org website is the official Blender Foundation source code snap shot builds of Blender from SVN. The builds provided here are built automatically periodically. These builds are built using Blender official features, and although not as stable as the Blender Official release builds, are often more stable than builds provided on graphicall.org. Because they are a snapshot of the most recent SVN trunk, they often have features which will only be available in the next official release of Blender. This gives the user the opportunity to test out and use new features before they become available in Blender Official releases.

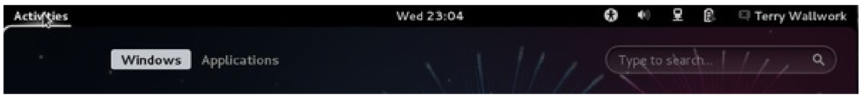
- builder.blender.org

Fedora based systems

Fedora Linux is an offshoot of the Redhat Linux distribution. Fedora Linux is used by Redhat to test new technologies which are eventually used within official Redhat Linux distributions. This means that Fedora Linux is a bleeding edge Linux distribution. This means that the libraries and software that are included with Fedora Linux are usually up to date, using some of the most recent versions of libraries and software available.

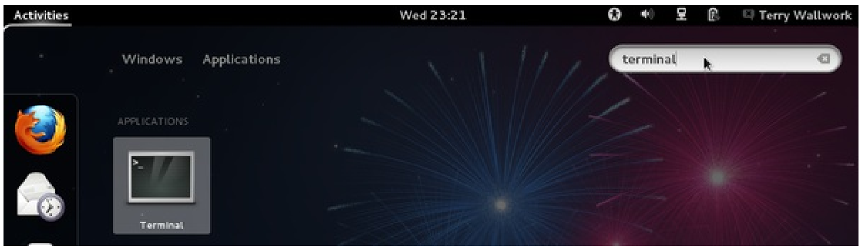
Opening a Terminal Window using Gnome Shell

Recent versions of Fedora Linux use the Gnome Shell desktop environment to interact with the user. To open a terminal window in Gnome Shell move your mouse pointer to the upper left corner of the screen and click on the Activities text.



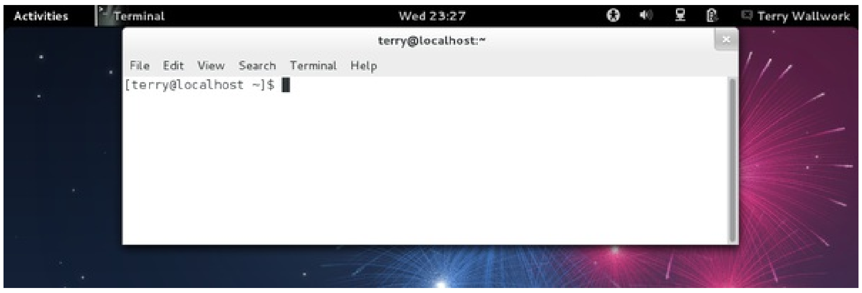
Mouse Pointer over Activities Area

Once you have clicked on the Activities text, move your mouse pointer to the upper right corner of the screen and click on the search field and type the word 'terminal' in the search field, then press ↵ Enter



Mouse Pointer over Search Area

This should result in a terminal window being opened on your desktop. Click on this window with your mouse pointer. At this point you should be able to type commands from the keyboard and they will be displayed in the terminal window.



```
[terry@localhost ~]$
```

Gnome Terminal Window

Installing Missing Blender dependencies with yum

Fedora Linux uses a package management frontend system called yum to install software packages and libraries.

If you have just recently installed a new or updated version of Fedora the first thing you should do is update your installed libraries and software. To do this, in your terminal window, type the following commands in the terminal window:

```
su root
```

You will then be asked to enter your root/admin password, enter this password. If you typed the root password correctly you will now be logged in as the root/admin user in that open terminal, which will mean you have enough permissions to install needed dependencies in Fedora Linux.

By default Fedora Linux has 1 missing library dependency which is required by Blender for it to run correctly. That missing library is the SDL library. To install that missing SDL library type the following command in the terminal window:

```
yum install SDL
```

Once the above command is typed, the yum package manager will ask for confirmation, type y at the terminal and press ↵ Enter:

```
Is this ok [y/N]:y
```

This will install the missing SDL library package.

Case Matters

It is important that you type SDL and not sdl, case matters.

You can now close the terminal as you will no longer need it.

Now that you have all the library dependencies installed to run Blender you can go to the Blender Download Website.

From the download page you can now choose the correct version of Blender to download for your particular hardware configuration.

Determining your Hardware Configuration

For Linux based systems such as Fedora Linux, Blender comes in 2 different versions a 32 bit version and 64 bit version. If you have a 32 bit computer platform you need to download and use the 32 bit version of Blender, otherwise you need to download the 64 bit version of Blender.

If you are not sure what sort of computer platform you are currently using you can determine weather you are running a 32 bit or 64 bit platform by opening a terminal window and typing the following command:

```
file /bin/cat
```

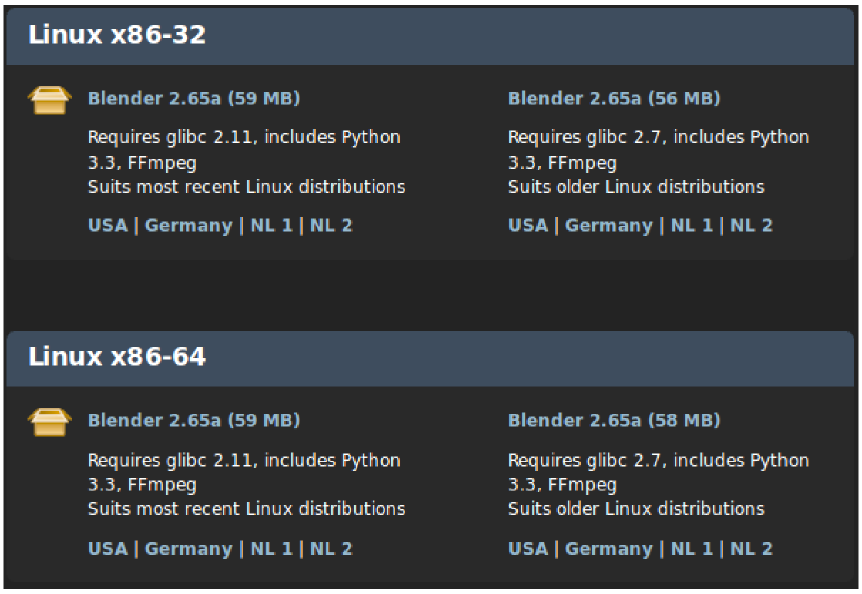
If the output of the above command starts with '/bin/cat: ELF 32-bit' you are using a 32 bit version of Fedora Linux and need to download a 32 bit version of Blender. If the output of the above command starts with '/bin/cat: ELF 64-bit' you are using a 64 bit version of Fedora Linux and need to download a 64 bit version of Blender.

32 bit on a 64 bit platform

If you are using a 64 bit version of Fedora Linux you can also use the 32 bit version of Blender, but doing so will mean you cannot use more than 4 gigabytes of memory, and the 32 bit version of Blender will run more slowly on a 64 bit Fedora Linux platform.

Downloading Blender From the Blender Download Website

Once you have determined which version of Blender you want to download, you can click on the corresponding link on the Blender Download Website.



Linux x86-32

Blender 2.65a (59 MB)
Requires glibc 2.11, includes Python 3.3, FFmpeg
Suits most recent Linux distributions
USA | Germany | NL 1 | NL 2

Blender 2.65a (56 MB)
Requires glibc 2.7, includes Python 3.3, FFmpeg
Suits older Linux distributions
USA | Germany | NL 1 | NL 2

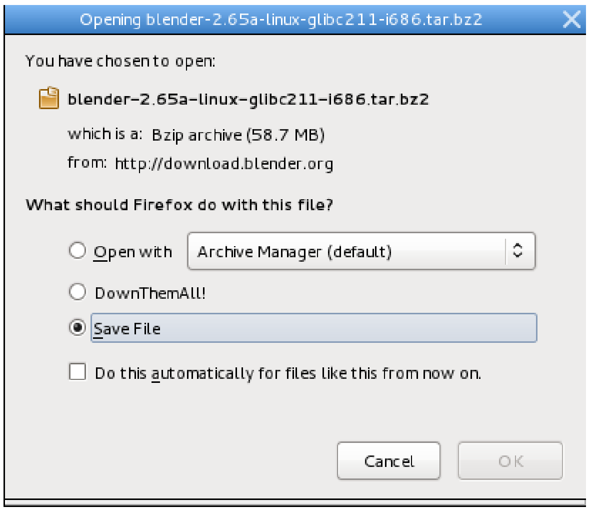
Linux x86-64

Blender 2.65a (59 MB)
Requires glibc 2.11, includes Python 3.3, FFmpeg
Suits most recent Linux distributions
USA | Germany | NL 1 | NL 2

Blender 2.65a (58 MB)
Requires glibc 2.7, includes Python 3.3, FFmpeg
Suits older Linux distributions
USA | Germany | NL 1 | NL 2

Blender Download Webpage

Once you do click on a link your web browser will possibly display a download dialog box asking you how you want to download Blender.



Opening blender-2.65a-linux-glibc211-i686.tar.bz2

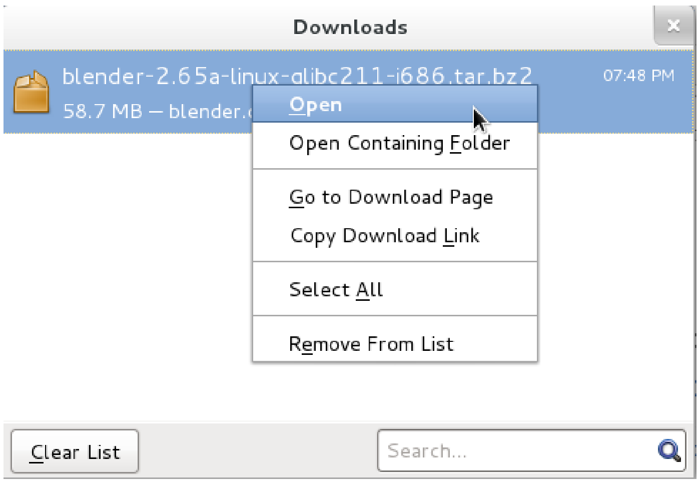
You have chosen to open:
blender-2.65a-linux-glibc211-i686.tar.bz2
which is a: Bzip archive (58.7 MB)
from: http://download.blender.org

What should Firefox do with this file?
○ Open with Archive Manager (default)
○ DownThemAll!
● Save File
☐ Do this automatically for files like this from now on.

Firefox File Download Dialog Box

In the file browser dialog box make sure the option 'Save File' is selected. Then click the OK button. This will download the Blender software to your Downloads directory.

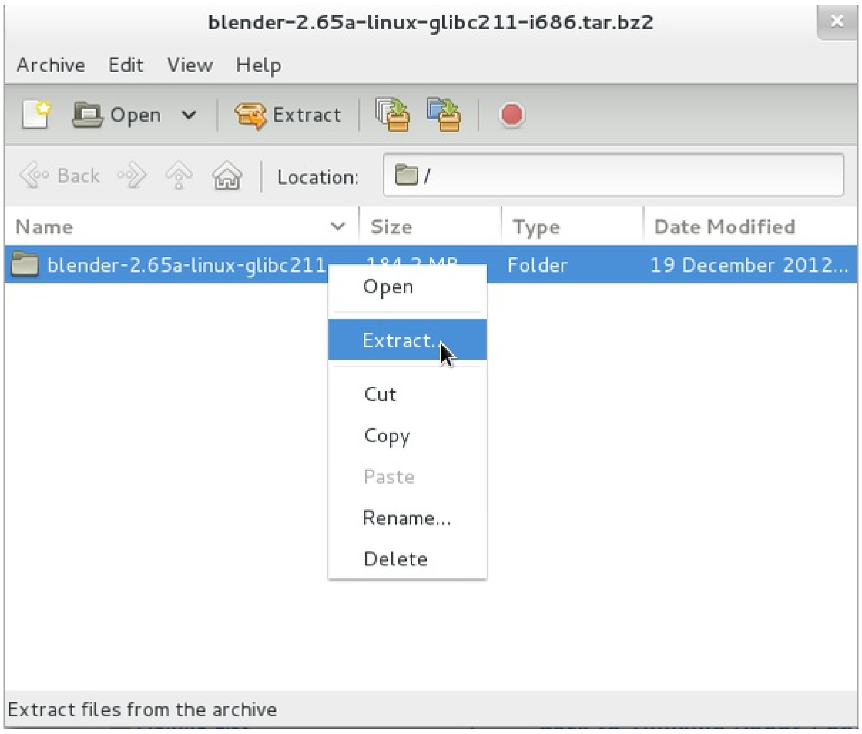
With your web browser window still selected press Ctrl+shift+y. This will open your browser download window. Right click on the Blender entry and select Open.



Firefox Download List Open

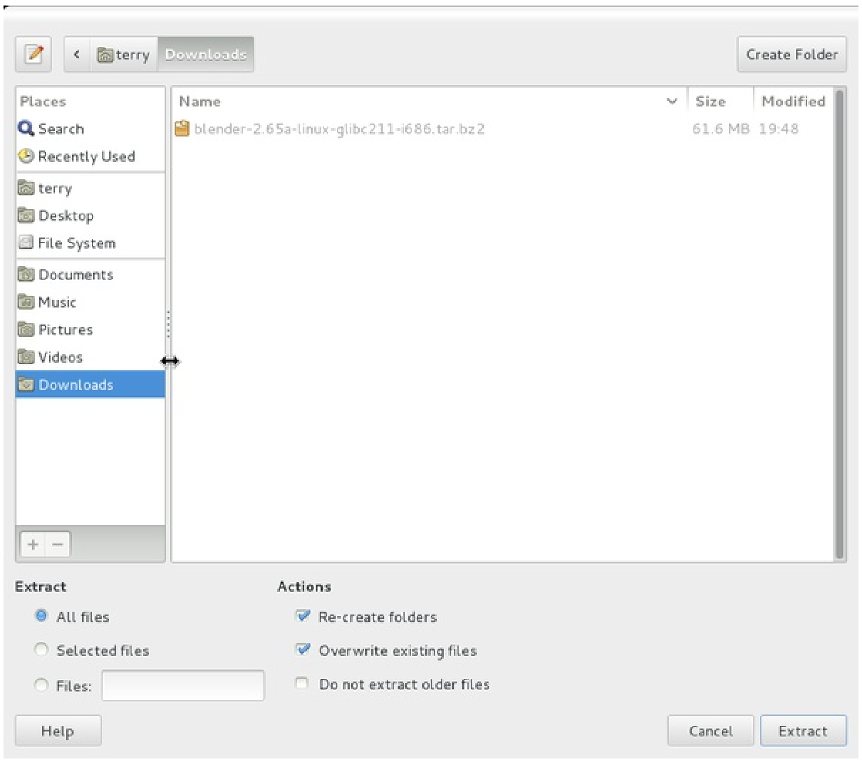
This will open the Blender software archive file in Fedora's default archive manager.

When the archive manager is displayed right click on the directory entry displayed in the archive manager and select the Extract entry from the popup menu that is displayed.



Archive Manager Extraction of Blender

Once the Extract entry is selected an Extract dialog box will be display, in this dialog box you can choose the location that you want to extract the Blender files to.



blender-2.65a-linux-glibc211-i686.tar.bz2 61.6 MB 19:48

Extract
- [✓ All files]
- Selected files
- Files:

Actions
- [✓ Re-create folders]
- [✓ Overwrite existing files]
- Do not extract older files

Archive Manager Extraction Location & Options

Make sure that in the Extract dialog box that the options All Files and Re-create Folders are both selected. Then you can press the Extract button and the Blender archive file will be extracted to whatever location you choose.

Once you have extracted the files from the Blender archive you will have a new directory at the location you extracted Blender to.

Executing Blender after it has been extracted

Once you have extracted Blender you can start Blender in a number of different ways:

- By opening a terminal window and then navigating to the directory Blender was extracted to:

```
cd ~/Download/blender-2.65a-linux-glibc211-i686
```

The above command would change into your home directory, from there it would change into your Downloads directory and from there it would change into the directory Blender was extracted to (in this case blender-2.65a-linux-glibc211-i686). Obviously if you extracted Blender to a different directory or are using a different version of Blender you would update the above command as appropriate.

Once you are in the directory the Blender binary is located in type the following command at the terminal

```
./blender
```

or

```
./blender-softwaregl
```

At this point if everything went well, you should see Blender displayed on screen.

Executing Blender In Hardware Or Software OpenGL Mode

Hardware or Software OpenGL Mode

There are 2 different ways of starting Blender. The first way is in Hardware Accelerated OpenGL mode, in this mode if your graphics card has Hardware support for OpenGL drawing commands Blender will use it. Blender will perform much more quickly when it is run in Hardware Accelerated OpenGL Mode. To start Blender in Hardware Accelerated OpenGL Mode type the following command at the terminal:

```
./blender
```

Some graphics cards either don't work at all or don't display information in Blender correctly when run this way. If this happens for you then you can run Blender in Software OpenGL Mode. To do this start Blender from the terminal by typing:

```
./blender-softwaregl
```

When started in this way Blender will use your CPU to process OpenGL drawing commands rather than using the dedicated hardware on your graphics card. This will result in Blender performing more slowly when doing 3D graphical tasks but it often will enable Blender to display correctly when it would not otherwise.

Operating System Keyboard Conflicts & Blender

Blender has a massive amount of keyboard shortcut keys that it uses and that are used very often by Blender users. Some of keyboard shortcuts that Blender uses however are also used by the Gnome Shell Window Manager. What follows is a list of the major conflicting keyboard shortcuts and how to change them.

Gnome Shell Window Manager Keyboard Shortcuts

Annoyingly the Gnome Shell Window Manager people have a habit of changing the way you alter the keyboard shortcut assignment. If you find that methods mentioned no longer work, please do a google search and you will find how to do it. The following commands work for Fedora 17/18 when using Gnome Shell Window Manager.

ALT+Left Mouse Button

Alt+lmb is a common keyboard shortcut used by Blender. It is also used by the Window Manager in Gnome Shell to move windows around. Because of this conflict using this keyboard shortcut to do edge loop selection does work as expected. To fix this issue you need to tell the Gnome Shell Window Manager not to use the keyboard short Alt+lmb. A common fix for this is to tell the Gnome Shell Window Manager to instead use Super+lmb. The Super key is also often called the Windows key.

To have Gnome Shell Window Manager use the Super key rather than Alt key when moving windows on the desktop, type the following command in a terminal window:

```
dconf write "/org/gnome/desktop/wm/preferences/mouse-button-modifier" "'<Super>'"
```

Obtaining Snapshot Versions of Blender

If you want to get versions of Blender which are more up to date as they are built from a current snapshot of the Blender SVN trunk periodically, you have a couple of websites you can use:

The graphicall.org website is a Blender users site where many different snap shots of Blender Source code are compiled by users and made available for download. This website has many builds of Blender with very experimental features enabled.

- www.graphicall.org

The builder.blender.org website is the official Blender Foundation source code snap shot builds of Blender from SVN. The builds provided here are built automatically periodically. These builds are built using Blender official features, and although not as stable as the Blender Official release builds, are often more stable than builds provided on graphicall.org. Because they are a snapshot of the most recent SVN trunk, they often have features which will only be available in the next official release of Blender. This gives the user the opportunity to test out and use new features before they become available in Blender Official releases.

- builder.blender.org

The if you want to build Blender from source code so you can get the latest greatest features of Blender, you can follow the official instruction. Building Blender from source is not difficult compared to trying to build other software of comparable complexity, but it takes some preparation and configuration to get right. If you take your time and read all the instructions, you should be able to do it.

- Official Blender Foundation Instruction For Building Blender From Source

If you still need help and have tried a google search then you can always goto the irc server irc.freenode.net #blendercoders channel and report the problem you are having. The coders are busy so they can take a while to help but they will do in general. If you don't have an irc client on your machine you can click the following link and that will connect you to irc through your web browser:

- irc.freenode.net #blendercoders channel

Being a Fedora user there's one more option for obtaining the latest development snapshot version of Blender from SVN. It comes in the form of a special script which automatically downloads all the source code and library dependencies that are required to build Blender directly from source code on a Fedora Linux system. This will only work for recent versions of Fedora, and has only been tested to work with 32 bit and 64 bit PC/Intel versions of Fedora (the script probably won't work for Mac computers). This is very very very very very experimental and temperamental and not official supported or condoned by the Blender Foundation script. But if you are a person who really wants to have a source compiled version of Blender and can't make sense of official instructions for building Blender from source, this script makes it slightly easier (when it works).

- AutoCompileBlender Script for Blender SVN Code

Enabling RPM Fusion Repository For Fedora

Fedora is an entirely open sourced operating system, it does not use any closed source software that is not released under some GPL type licence. This means that some important features and software such as codecs, libraries and drivers are not provided by the Fedora Project.

To get around some of these limitations a software repository was setup that is external to the official Fedora Project. The RPM Fusion Repository provides lots of extra software which contain software that does not meet the licensing standards required of the Fedora Project.

Some features of Blender require certain libraries and features that are only provided in the RPM Fusion Repository, and so you need to install and enable the RPM Fusion Repository for your Fedora operating system.

Go to the RPM Fusion Repository website and follow the instructions on how to install and enable the RPM Fusion Repository for your Fedora Linux system.

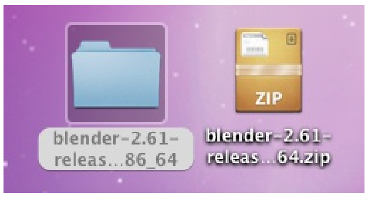
- RPM Fusion Repository Website

Installing CUDA Support In Fedora For Blender GPU Cycles Rendering Support

Yet to be written researching how to do this for fedora. I can't test because I don't have a GPU based compatible card.

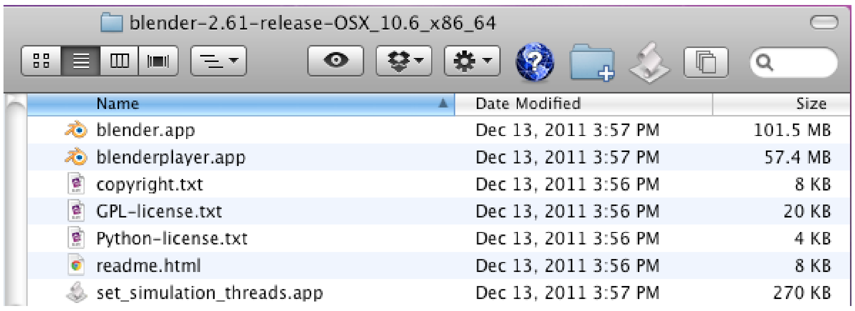
Installing on Mac

Installing Blender on a Mac is very easy. There are pre-compiled versions for 32 and 64 Bit Macs using Intel Processors. Once you know your platform and processor :



Blender Unzipped (left) and Zipped (right)

- Download the appropriate Blender for your system from blender.org download page
- Click/Right Click RMB in the Downloaded file, choose *Unzip to a Folder*.
- A Folder where you downloaded Blender will be created, with the same name of the downloaded file, without extension.
- Click in the folder, it will be opened in Finder (See Fig: Opening Blender From Finder - Below)
- Click twice LMB in *blender.app* file. You're ready to go !


blender-2.61-release-OSX_10.6_x86_64

Name	Date Modified	Size
blender.app	Dec 13, 2011 3:57 PM	101.5 MB
blenderplayer.app	Dec 13, 2011 3:57 PM	57.4 MB
copyright.txt	Dec 13, 2011 3:56 PM	8 KB
GPL-license.txt	Dec 13, 2011 3:56 PM	20 KB
Python-license.txt	Dec 13, 2011 3:56 PM	4 KB
readme.html	Dec 13, 2011 3:56 PM	8 KB
set_simulation_threads.app	Dec 13, 2011 3:57 PM	270 KB

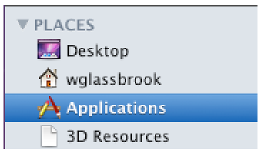
Opening Blender from Finder - Example using Blender 2.61 for 64 bits Intel Mac, Mac OS X 10.6

Supported Platforms for Blender 2.6X and above

Since 2.63, we have no more pre-compiled versions for PPCs G5 (PowerMacs).
Blender 2.64 Versions and above will only run on Macs using Intel processors, Mac OS X 10.6 or higher is required.

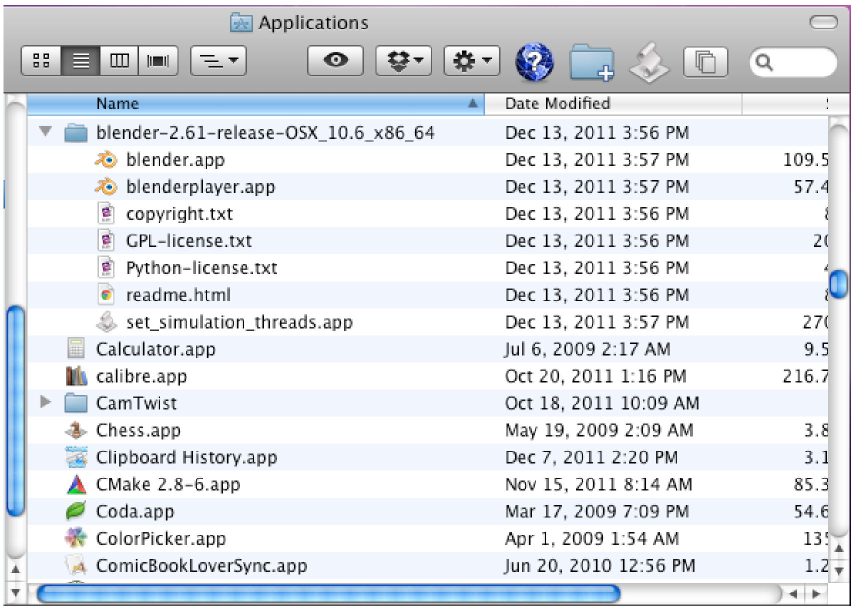
Important: If you need to run Blender from a Command Line to see the Console Window, visit the page about the Blender Console Window

Adding Blender to Applications



Places and Applications on Mac

You can move Blender to the Mac OS X *Applications* folder, it will work like any other application in your Mac. You can do this copying or moving the unzipped folder to the Mac *Applications* folder. Find the *Applications* folder, using the Finder File Browser and searching for *Places*. At the left side, opening the *Places* Tab, you will find your *Applications* folder. Move the Unzipped Blender Folder to the Applications.



Name	Date Modified	
blender-2.61-release-OSX_10.6_x86_64	Dec 13, 2011 3:56 PM	
blender.app	Dec 13, 2011 3:57 PM	109.5
blenderplayer.app	Dec 13, 2011 3:57 PM	57.4
copyright.txt	Dec 13, 2011 3:56 PM	8
GPL-license.txt	Dec 13, 2011 3:56 PM	20
Python-license.txt	Dec 13, 2011 3:56 PM	4
readme.html	Dec 13, 2011 3:56 PM	8
set_simulation_threads.app	Dec 13, 2011 3:57 PM	270
Calculator.app	Jul 6, 2009 2:17 AM	9.5
calibre.app	Oct 20, 2011 1:16 PM	216.7
CamTwist	Oct 18, 2011 10:09 AM	
Chess.app	May 19, 2009 2:09 AM	3.8
Clipboard History.app	Dec 7, 2011 2:20 PM	3.1
CMake 2.8-6.app	Nov 15, 2011 8:14 AM	85.3
Coda.app	Mar 17, 2009 7:09 PM	54.6
ColorPicker.app	Apr 1, 2009 1:54 AM	135
ComicBookLoverSync.app	Jun 20, 2010 12:56 PM	1.2

Blender, when moved to Applications folder

Adding Blender to your Dock

To add Blender to your dock, you can do the following:

- When Blender is placed in your *Applications* folder, click in *blender.app* and drag it to your dock.
- While Blender is running, right click RMB on the Icon present in your Mac Dock, choose *Keep in Dock*.

Installing on Windows

Download

You can obtain the latest stable version of Blender for Windows from the Blender download page.

Version

Blender for Windows is currently available in 32-bit and 64-bit versions. Users with a 32-bit version of Windows must download the 32-bit version of Blender. Users with a 64-bit version of Windows can choose to use either the 32-bit or 64-bit version of Blender, however you will likely notice an increase in performance when using the 64-bit version of Blender, especially on systems with large amounts of RAM.

To identify whether your Windows Vista or Windows 7 computer is 32- or 64-bit, you can:

- Click the Windows Icon start button, right click the **Computer** options and then click **Properties**.
- The system type (32- or 64-bit) will be displayed under the **System** option.

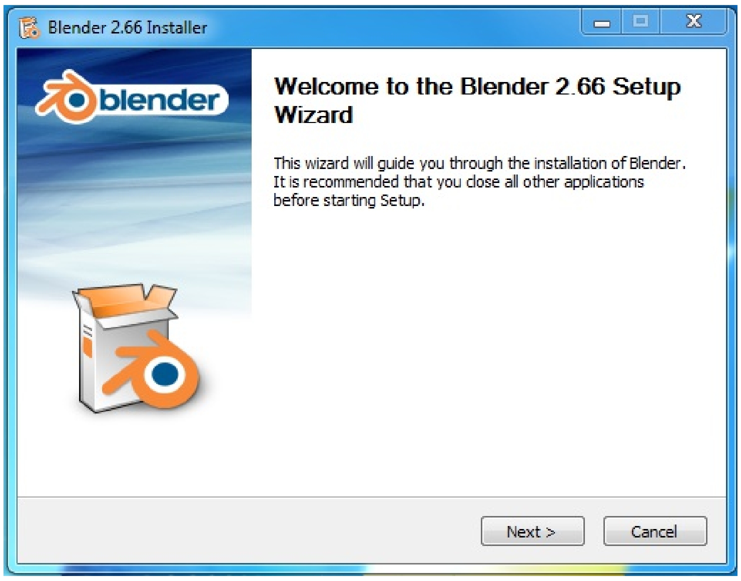
If you are using Windows XP:

- Click **Start**.
- Right-click **My Computer**
- Click **Properties.**
- If you are using the 64-bit version of Windows XP, *x64 Edition* will be displayed under the **System** option.

Installation

Once your download has finished, navigate to the *Downloads* folder and double-click the Blender executable file to start the installation process. Keep in mind that installing Blender requires Administrator rights.

Welcome screen

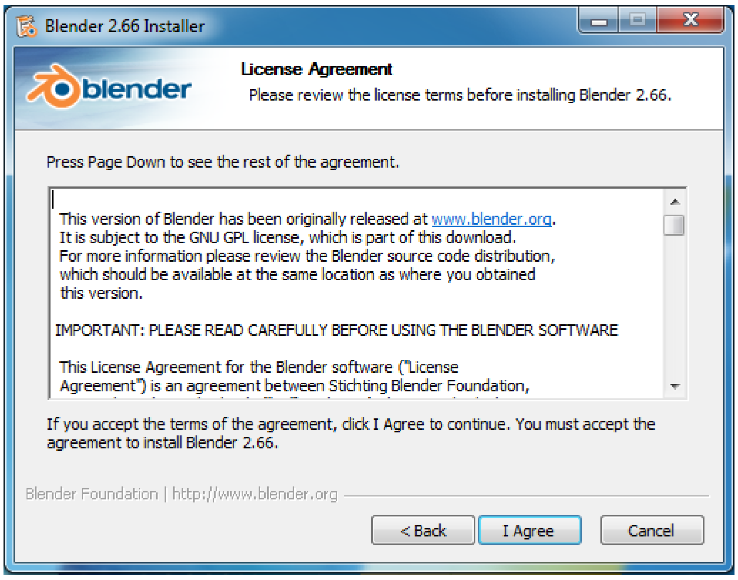


Welcome to the Blender 2.66 Setup Wizard

This wizard will guide you through the installation of Blender.
It is recommended that you close all other applications before starting Setup.

The Welcome is the first screen of the installation process. Click Next to continue.

License agreement


Blender 2.66 Installer

License Agreement

Please review the license terms before installing Blender 2.66.

Press Page Down to see the rest of the agreement.

This version of Blender has been originally released at www.blender.org.
It is subject to the GNU GPL license, which is part of this download.
For more information please review the Blender source code distribution,
which should be available at the same location as where you obtained
this version.

IMPORTANT: PLEASE READ CAREFULLY BEFORE USING THE BLENDER SOFTWARE

This License Agreement for the Blender software ("License
Agreement") is an agreement between Stichting Blender Foundation,

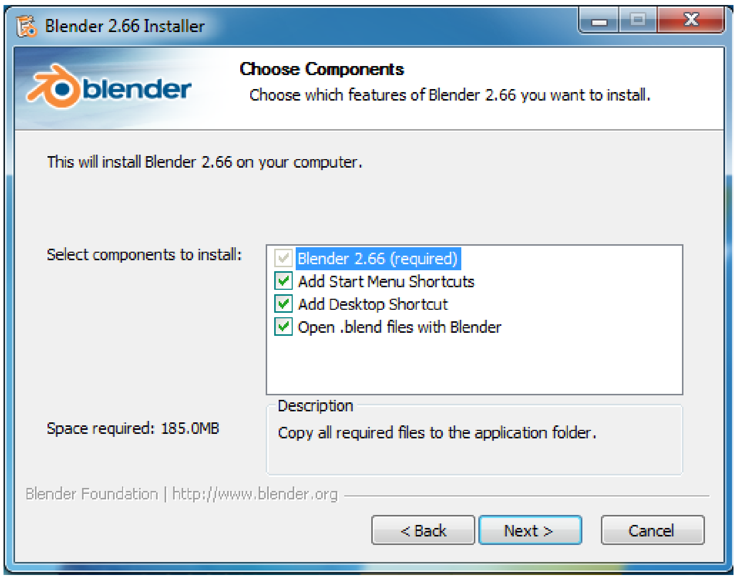
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Blender 2.66.

Blender Foundation | http://www.blender.org

The License agreement must be accepted before the installation will continue.

Installation options

Program options


Blender 2.66 Installer

Choose Components

Choose which features of Blender 2.66 you want to install.

This will install Blender 2.66 on your computer.

Select components to install:

- [✓] Blender 2.66 (required)
- [✓] Add Start Menu Shortcuts
- [✓] Add Desktop Shortcut
- [✓] Open .blend files with Blender

Space required: 185.0MB

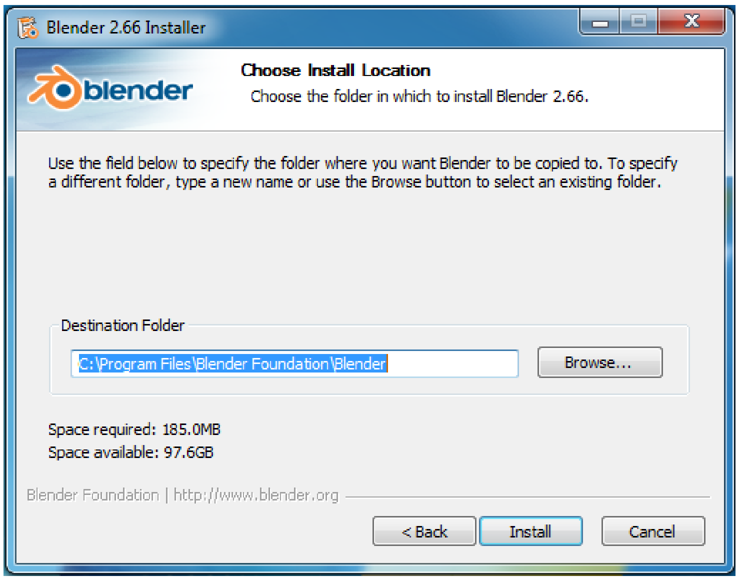
Description

Copy all required files to the application folder.

Blender Foundation | http://www.blender.org

Select any desired options.

Location


Blender 2.66 Installer

Choose Install Location

Choose the folder in which to install Blender 2.66.

Use the field below to specify the folder where you want Blender to be copied to. To specify a different folder, type a new name or use the Browse button to select an existing folder.

Destination Folder

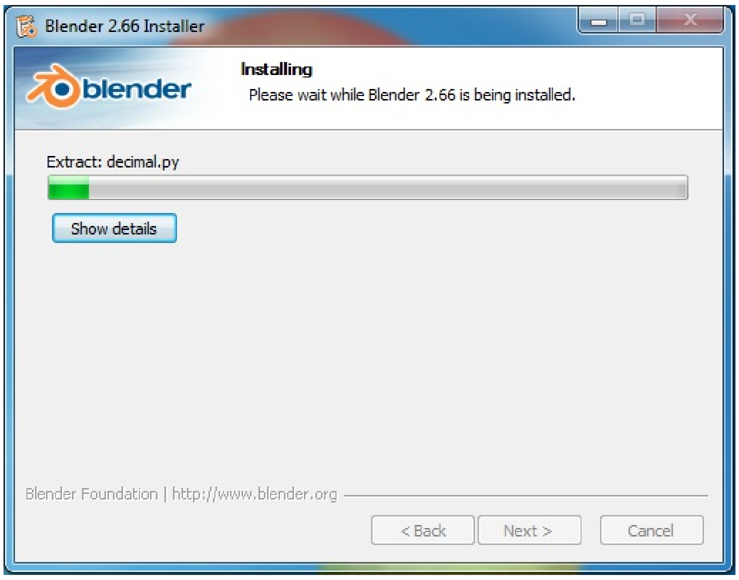
C:\Program Files\Blender Foundation\Blender

Space required: 185.0MB
Space available: 97.6GB

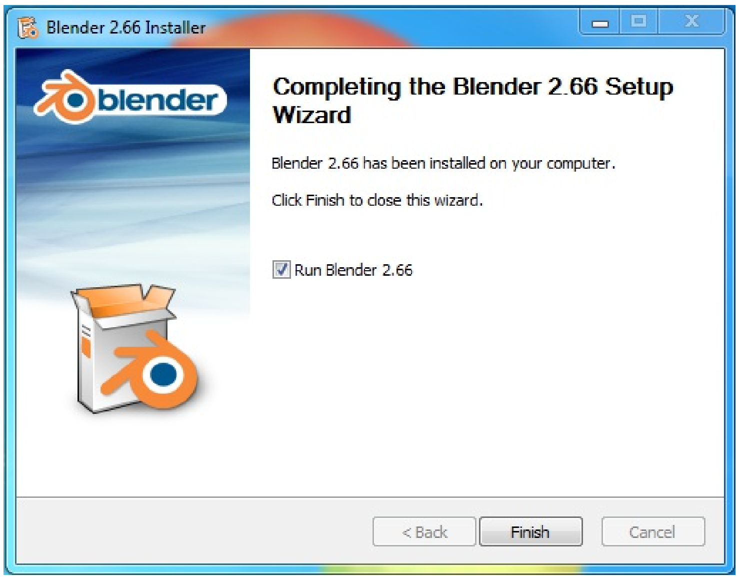
Blender Foundation | http://www.blender.org

Here you can choose the directory where Blender will be installed.

Installing


Blender 2.66 Installer
Installing
Please wait while Blender 2.66 is being installed.
Extract: decimal.py
Show details
Blender Foundation | http://www.blender.org
< Back
Next >
Cancel

Installed


Blender 2.66 Installer
Completing the Blender 2.66 Setup Wizard
Blender 2.66 has been installed on your computer.
Click Finish to close this wizard.
Run Blender 2.66
< Back
Finish
Cancel

And there you go! A fresh installation of Blender 2.5. After enabling or disabling the option to run Blender right after closing the installation, click 'Finish'.

Portable Install

If you want to run Blender off a USB key so that you can use it wherever you go, download the .zip version and extract it to a USB key. You may want to avoid storing animation output or other temporary files on the USB key as frequent drive writes may shorten its life.

To keep all configuration files and installed addons on the USB key, create a folder named *config* in the unzipped Blender folder. Now all configuration files will be read from and written to the USB key rather than the computer you're running Blender on.

Other supported operating systems

FreeBSD

Download the file `blender-2.##-FreeBSD-####.tbz` from the Blender download page where `2.##` is the Blender version and `####` is the machine architecture (i386 or amd64).

To start Blender

- Unpack the archive
- Open a shell and navigate to the unpacked archive's directory
- Execute the command `./blender` while the X.Org server is running.

Post Install Configuration

Blender will normally run fine without _any_ configuration but there are some changes you may want to make depending on your hardware.

This section **only** covers system specific user preferences.

Info

The key combination CtrlU saves all the settings of the currently open Blender file into the default Blender file. The settings in the default Blender file are read when Blender is first started or when CtrlN is pressed to start a new file. If you accidentally change the settings in your default Blender file (i.e. you save your work in progress as the default) you can restore factory settings from the file menu press CtrlU to save the newly loaded factory settings.

The following section titles match the user preference categories.

Input

- If you don't have a numpad you may want to enable 'Emulate Numpad'
- If you don't have a middle mouse button you can enable 'Emulate 3 Button Mouse'

File Paths

This isn't essential but you may want to set the paths for more useful default locations.

- Temp Directory: You may want to change this if you have a faster disk for temp file storage.
- Image Editor: Useful so you can edit images externally (from the image space), The path to the gimp for example can be set here.

Info

"//" at the start of a path in blender signifies the directory of the currently opened blend file, used to reference relative-paths.

System

While there are many settings here you may want to set in special-cases, this document focuses on common features.

- VBOs (Vertex Buffer Objects), can give a good speed boost to the viewport performance, keep this enabled unless it gives problems (some older hardware does not support VBOs)

Sequencer / Clip Editor:

- Memory Cache Limit: If you use the sequencer of movie clips you may want to increase the cache limit since this will make scrubbing a lot faster - but take care the limit stays well under your total system memory.

Configuration & Data Paths

Blender can be installed system wide or run from an extracted bundled with all necessary files contained.

There are 3 different directories blender may use, their exact locations are operating system dependent.

- **LOCAL:** location of configuration and runtime data (for self contained bundle)
- **USER:** location of configuration files (normally in the user's home directory).
- **SYSTEM:** location of runtime data for system wide installation (may be read-only).

For system installations both **SYSTEM** & **USER** directories are needed.

For locally extracted blender distributions the user configuration and data runtime data are kept in the same sub-directory, allowing multiple blender versions to run without conflict, ignoring the **USER** and **SYSTEM** files.

Here are the default locations for each system:

OSX

LOCAL: `./2.71/`
USER: `/Users/{user}/Library/Application Support/Blender/2.71/`
SYSTEM: `/Library/Application Support/Blender/2.71/`

note, OSX stores blender binary in ./blender.app/Contents/MacOS/blender, so the local path to data & config is ./blender.app/Contents/MacOS/2.71

Windows

LOCAL: `.\2.71\`
USER: `C:\Documents and Settings\{username}\AppData\Roaming\Blender Foundation\Blender\2.71\`
SYSTEM: `C:\Documents and Settings\All Users\AppData\Roaming\Blender Foundation\Blender\2.71\`

Unix (Linux/BSD/Solaris)

LOCAL: `./2.71/`
USER: `$HOME/.config/blender/2.71/`
SYSTEM: `/usr/share/blender/2.71/`

note that ./2.71/ is relative to the blender executable & used for self contained bundles distributed by official blender.org builds.
note the USER path will use XDG_CONFIG_HOME if its set: `$XDG_CONFIG_HOME/blender/2.71/`

Path Layout

This is the path layout which is used within the directories described above.

Where ./config/**startup.blend** could be ~/.blender/2.66/config/startup.blend for example.

- `./autosave/ ...`
  autosave blend file location. *Windows only, temp directory used for other systems.*
  search order: **LOCAL, USER**

- `./config/ ...`
  defaults & session info
  search order: **LOCAL, USER**
- `./config/`**`startup.blend`**
  default file to load on startup.
- `./config/`**`userpref.blend`**
  default preferences to load on startup.
- `./config/`**`bookmarks.txt`**
  file selector bookmarks.
- `./config/`**`recent-files.txt`**
  recent file menu list.

- `./datafiles/ ...`
  runtime files
  search order: **LOCAL, USER, SYSTEM**
- `./datafiles/locale/{language}/`
  Static precompiled language files for UI translation.
- `./datafiles/icons/*.png`
  icon themes for blenders user interface. *not currently selectable in the theme preferences.*
- `./datafiles/brushicons/*.png`
  images for each brush.

- `./scripts/ ...`
  python scripts for the user interface and tools
  search order: **LOCAL, USER, SYSTEM**

- `./scripts/addons/*.py`
  python addons which may be enabled in the user preferences, includes import/export format support, render engine integration and many handy utilities.
- `./scripts/addons/modules/*.py`
  modules for addons to use (added to pythons sys.path)
- `./scripts/addons_contrib/*.py`
  another addons directory which is used for community maintained addons (must be manually created).
- `./scripts/addons_contrib/modules/*.py`
  modules for addons_contrib to use (added to pythons sys.path)
- `./scripts/modules/*.py`
  python modules containing our core API and utility functions for other scripts to import (added to pythons sys.path)
- `./scripts/startup/*.py`
  scripts which are automatically imported on startup.
- `./scripts/presets/{preset}/*.py`
  presets used for storing user defined settings for cloth, render formats etc.
- `./scripts/templates/*.py`
  example scripts which can be accessed from: Text Space's Header -> Text -> Script Templates

- `./python/ ...`
  bundled python distribution only necessary when the systems python is absent or incompatible
  search order: **LOCAL, SYSTEM**

Notes

User Scripts Path

The user preferences script path provides a way to set your own directory which is used for scripts as well as the user scripts path. Be sure to create subfolders within this directory which match the structure of blenders scripts directory, startup/, addons/, modules/ etc. because copying scripts directly into this folder will not load them on startup or as addons.

Environment Variables

Environment variables can be used to override default path locations, eg: $BLENDER_USER_CONFIG, $BLENDER_SYSTEM_PYTHON.

This is not normally something which needs setting but can be useful for custom configurations.

For details see the 'Environment Variables' section in 'blender --help'

Scripts Path & Missing Buttons

If blender starts with no interface this is probably because the scripts are not loading correctly and can be caused by...

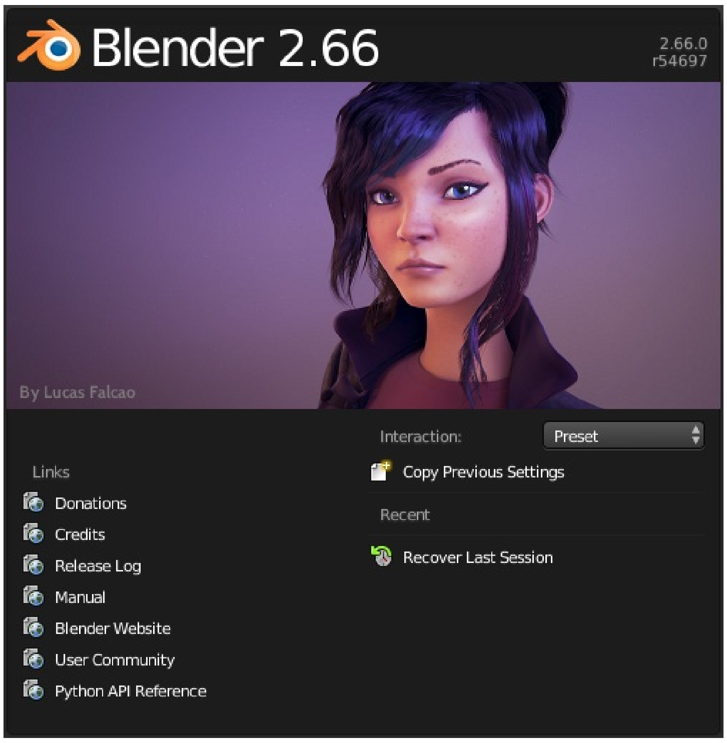
- script path not found.
- an error in one of the scripts.
- a version mis-match between blender and the scripts.

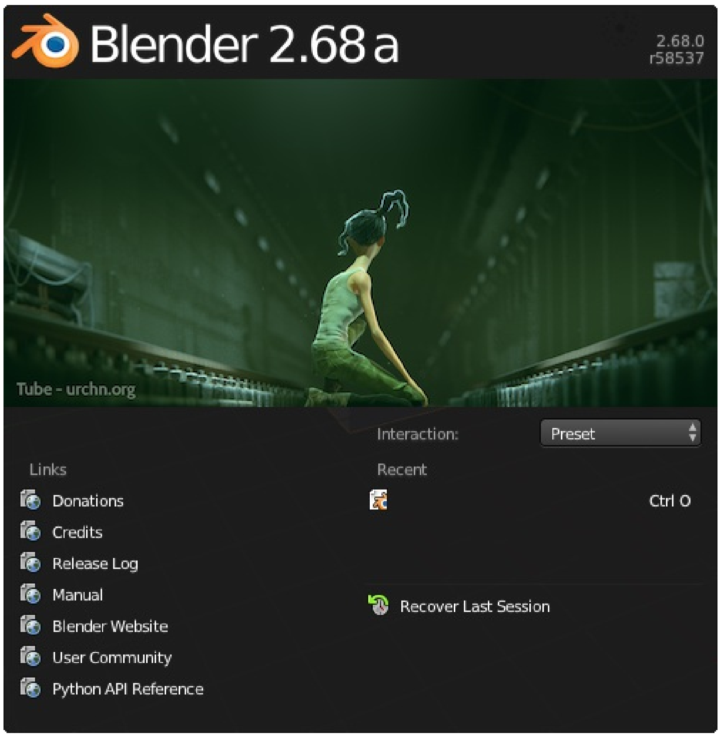
It's best to load blender from a terminal to see any error messages to see what's wrong.

Starting Blender for the first time

If you are familiar with Blender 2.4x or other 3D software such as Maya, 3ds Max or XSI, you will immediately notice that Blender is quite different from what you are used to seeing. However you will soon see similarities with your previous software, like a 3D Viewport, an Outliner and a Timeline. If this is the first time you have used any 3D software, you may be a little lost. Fortunately there's really only one rule when you want to learn 3D with Blender: don't be afraid to explore and experiment!

After starting Blender, take a look at the splash screen where you will see the Blender version in the top right-hand corner.





The left side shows you some useful links like the release log of the version you are using (what's new in this version), the wiki manual (what you're reading now) and the official Blender website. These links are also accessible from the Help menu.

The right side lists recent blender files (.blend) you have saved. If you're running Blender for the first time, this part will be empty. This list is also available in File » Open Recent. The interaction menu lets you choose a keymap preset (by default, Blender or Maya) are available.

To start using Blender, you have three options:

- Click on one of the recent files (if you have any)
- Click anywhere else on the screen (except the dark area of the splash screen) or

- Press Esc to start a new project

Save your work regularly

Blender does not warn you of any unsaved data when you exit the program, so remember to save often! If you do close Blender without saving your last actions, all is not lost. Just open Blender again and click on Recover Last Session in the Splash Screen. You also have this option in the main menu via File » Recover Last Session.

Temporary .blend file
Every time Blender exits, it saves the current data in a temporary .blend file. When you recover your last session, Blender will load the data from that file.

Interface concepts

Blender is developed as cross-platform software which means that its primary target is to work seamlessly in all major operating systems, including Linux, Mac OS X and Windows.(1)

Since the Blender interface is based on OpenGL, you will find that it is consistent between the major operating systems.

(1) Other operating systems are supported by third party developers through source compilation.

The 3 Rules

The Blender user interface is based on 3 main principles:

- **Non Overlapping**: The UI permits you to view all relevant options and tools at a glance without pushing or dragging windows around(2).
- **Non Blocking**: Tools and interface options do not block the user from any other parts of Blender. Blender doesn't pop up requesters that require the user to fill in data before things execute.
- **Non Modal**: User input should remain as consistent and predictable as possible without changing commonly used methods (mouse, keyboard) on the fly.

(2)However, Blender 2.5 permits multiple windows for multi-screen setup. It is an exception to the *Non overlapping rule*.

Powerful interface

Blender's interface is drawn entirely in OpenGL which allows you to customize your interface to suit your needs. Windows and other interface elements can be panned, zoomed and their content moved around. Once your screen is organized exactly to your taste for each specialized task it can then be named and saved.

Blender also makes heavy use of keyboard shortcuts to speed up your work. The keymaps can be edited to make memorizing them easier.

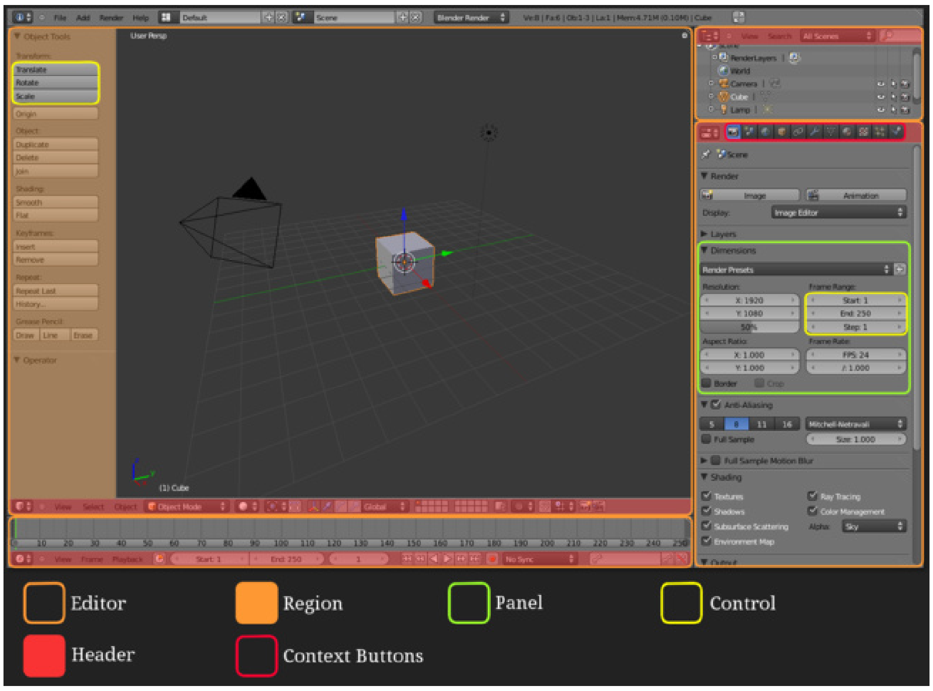
Overview

Let's have a look at the default interface. It is composed of Editors, Headers, Context buttons, Regions, Panels and Controls.

- In Blender, we call an **Editor** the parts of the software which have a specific function (3D view, Properties Editor, Video Sequence Editor, Nodes Editor...). Each editor has its own *Header* at the top or bottom.
- **Context buttons** give access to options. They are like tabs and are often placed on an editor header (like Properties Editor).
- For each editor, options are grouped in **Panels** to logically organize the interface (Shadow panel, Color panel, Dimensions panel...).
- **Regions** are included in some editors. In that case, panels and controls are grouped there. For workspace optimization, it is possible to temporarily hide regions with the hotkeys T and N for the Toolbar and Properties Region respectively.

- Panels contain **Controls**. These can let you modify a function, an option, or a value. In Blender, there are several types of controls:
    - **Buttons**: Permit access to a tool (Translate, Rotate, Insert Keyframe). These tools usually have a keyboard shortcut to speed up your work. To display the shortcut, just hover your mouse over a button to see the tooltip.
    - **Checkboxes**: Permit enabling or disabling of an option. This control can only contain a boolean value (True/False, 1/0).
    - **Sliders**: Allows you to enter floating values. These can be limited (e.g. from 0.0 to 100.0) or not (e.g. from -∞ to +∞). Notice that two types of sliders exist in Blender.
    - **Menus**: Permits a value to be chosen from a list. The difference between this and a Checkbox is that values are named and there can be more than two values on these menus.

Read more about buttons and controls »



General Usage of Input Devices

Blender's workflows are optimized for parallel usage of mouse and keyboard. That's why we have evolved the 'golden rule':

Keep one hand on the mouse and the other on the keyboard

The most frequently used keys are grouped so that they can be reached by the left hand in standard position (index finger on F) on the English keyboard layout. This assumes that you use the mouse with your right hand.

Input configuration

Blender's interface is designed to be used with the following recommended input configuration:

- A three-button mouse with a scroll wheel
- A full keyboard with a numeric keypad (NumLock should generally be switched on).

Read more about Blender configuration »

Usage of Mouse Buttons

The mouse takes an important role while working with Blender. Therefore we have established a general usage of the mouse buttons which apply in most cases:

Right Mouse Button: Select an item
Left Mouse Button: Initiate or confirm an action

(These can be swapped in the Input tab of the User Preferences)

Note: The Reference section contains details (and a short video guide) about the Usage of Mouse Buttons for basic Operations

Conventions in this Manual

This manual uses the following conventions to describe user input:

- The mouse buttons are referred to as:

  LMB - Left Mouse Button
  MMB - Middle Mouse Button
  RMB - Right Mouse Button

- If your mouse has a wheel:

  MMB - Middle Mouse Button (clicking the wheel)
  Wheel - rolling the wheel.

- Hotkey letters are shown in this manual like they appear on a keyboard; for example,

  G - refers to the lowercase "g".

  ⇧ Shift, Ctrl and Alt are generally specified as modifier keys

  CtrlW or ⇧ ShiftAltA - indicates that these keys should be pressed simultaneously

  0 NumPad to 9 NumPad, + NumPad - and so on refer to the keys on the separate numeric keypad.

Other keys are referred to by their names, such as Esc, ⇆ Tab, F1 to F12. Of special note are the arrow keys, ←, → and so on.

Mouse Button Emulation

If you do not have a 3 button mouse, you'll need to emulate it by checking the option in the User Preferences (unchecked by default).

The following table shows the combinations used:

3-button Mouse	2-button Mouse	Apple Mouse
LMB	LMB	LMB (mouse button)
MMB	Alt LMB	⌥ Opt LMB (Option/Alt key + mouse button)
RMB	RMB	⌘ Cmd LMB (Command/Apple key + mouse button)

All the Mouse/Keyboard combinations mentioned in the Manual can be expressed with the combinations shown in the table. For Example, ⇧ ShiftAlt RMB becomes ⇧ ShiftAlt⌘ Cmd LMB on a single-button mouse.

NumPad Emulation

If you do not have a Numeric Numpad on the side of your keyboard, you may want to Emulate one (uses the numbers at the top of the keyboard instead, however removes quick access to layer visibility).

Read more about NumPad Emulation on User Preferences page »

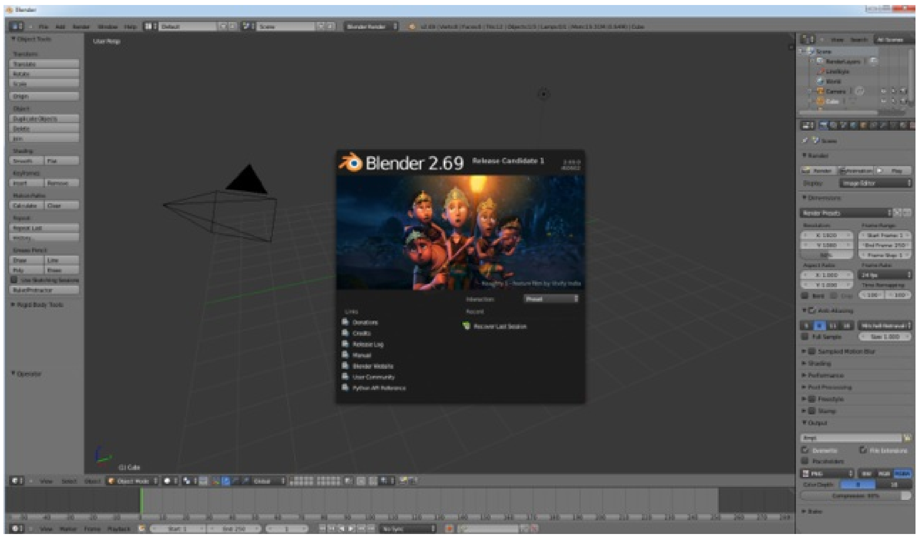
Non English Keyboard

If you use a keyboard with a non-english keyboard layout, you still may benefit from switching your computer to the UK or US layout as long as you work with Blender. Note that you can also change the Blender default keymap and change the default hotkeys. However this manual is based on the default keymap.

Read more about Blender configuration »

The Window System

When you start Blender you should see a screen similar to this (the splash screen in the center will change with new versions):



In the center of the window is the splash screen. This gives quick and easy access to recently opened Blender files. If you want to start work on a new file just click outside of the splash screen. The splash screen will disappear revealing the default layout and cube.

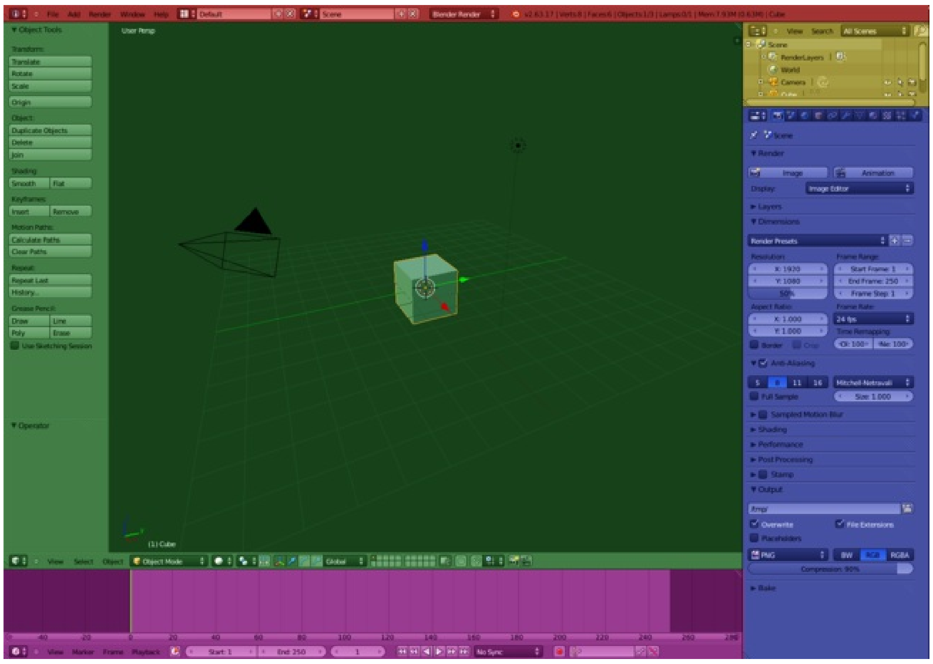
Every window you see can be further broken down into separate areas (as described in the section on arranging frames). The default scene is described below.

The default scene

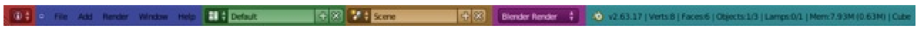
The default scene is separated into five windows and is loaded each time you start Blender or a new file. The five windows are:

- The Info window (shaded red) at the top. The Info window is comprised solely of a header.
- A large 3D window (3D View) (shaded green).
- A Timeline window at the bottom (shaded purple).
- An Outliner window at the top right (shaded yellow).
- A Properties window (Buttons window) at the bottom right (shaded blue).

As an introduction we will cover a few of the basic elements.


Default Blender scene and Window arrangement

The Info Window (main menu)

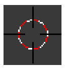

Info Window

The Info Window is found at the top of the Default Scene and has the following components:

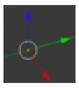
- **Window/Editor Type Selector**: The red shaded area allows you to change the Window/Editor Type. This region is found on every Window.

- **Menu options**: The dark blue shaded area provides access to the main menu options.
- **Current Screen (default is Default)**: The green shaded area allows you to select different Screens. By default, Blender comes with several pre-configured Screens for you to choose from. If you need custom screen layouts, you can create and name them.
- **Current Scene**: The yellow shaded area allows you to select different Scenes. Having multiple Scenes allows you to work with separate virtual environments, with completely separate data, or with objects and/or mesh data linked between them. (In some 3D packages, each file contains one scene, while in Blender, one .blend file may contain several scenes.)
- **Current Engine**: The purple shaded area gives a list of available rendering and game engines.
- **Resource Information**: The aqua shaded area gives you information about Blender and system resources in use. This region will tell you how much memory is being consumed based on the number of vertices, faces and objects in the selected scene, as well as totals of what resources are currently selected. This can help identify when you are reaching the limits of your hardware.

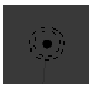
3D Window View



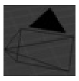
- **3D Cursor**: Can have multiple functions. For example, it represents where new objects appear when they are first created, or it can represent where the center of a rotation will be.



- **3D Transform Manipulator**: Is a visual aid in transforming objects (grab/move, rotate and scale). Objects can also be transformed using the keyboard shortcuts: (G/R/S); CtrlSpace will toggle the manipulator visibility.
- **Cube Mesh**: By default, a new installation of Blender will always start with a Cube Mesh sitting in the center of Global 3D space (in the picture above, it has been moved). After a while, you will most likely want to change the "Default" settings; this is done by configuring Blender as you would want it on startup and then saving it as the "Default" using CtrlU (Save Default Settings).

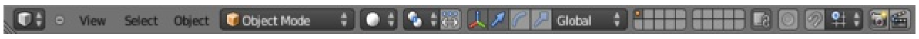


- **Light (of type Lamp)**: By default, a new installation of Blender will always start with a Light source positioned somewhere close to the center of Global 3D space.



- **Camera**: By default, a new installation of Blender will always start with a Camera positioned somewhere close to the center of Global 3D space and facing it.

3D Window Header



3D Window Header

This is the header for the 3D window. All windows in Blender have a header, although in some cases they may be located at bottom of the window.

Read more about Blender headers »

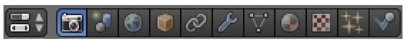
- **Window/Editor Type Selector**: Allows you to change the type of Window. This option can be found in every window header. For example, if you want to see the Outliner window you would click and select it.

- **3D Transform manipulator options**: Access to the manipulator widget is also possible by clicking the coordinate system icon on the toolbar. The translation/rotation/scale manipulators can be displayed by clicking each of the three icons to the right of the coordinate system icon. ⇧ Shift LMB -clicking an icon will add/remove each manipulator's visibility.

- **Viewport shading**: Blender renders the 3D window using OpenGL. You can select the type of Viewport shading that takes place by clicking this button and selecting from a variety of shading styles including simple bounding boxes and complex textures. It is recommended that you have a powerful graphics card if you are going to use the Textured style.

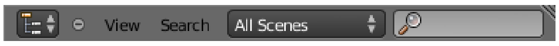
- **Layers**: Blender Layers are provided to help distribute your objects into functional groups. For example, one layer may contain a water object and another layer may contain trees, or one layer may contain cameras and lights. To de-clutter the view you can turn layers on and off.

Buttons (Properties) Window Header


Properties Window Header

The Properties window displays panels of functions. Panels that contain similar functions are grouped, e.g. all of the rendering options are grouped. In the header of the Properties Windows is a row of buttons (called Context Buttons) that allow you to select which group of panels are shown. Some panels are only visible when particular Objects are selected. Panels can be collapsed by use of the small arrow left of the panel title (e.g. besides *Render*) and may be rearranged by dragging the top right corner.

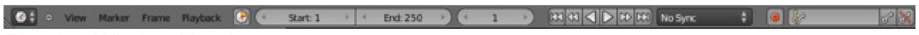
Outliner Window



Outliner Window Header

This window lists all the objects in a scene and can be very useful when working with larger scenes with lots of items. You can choose what types of elements and how they are displayed in the header.

Timeline Window



Timeline Window Header

This window gives a timeline, through which you can scrub with the LMB.

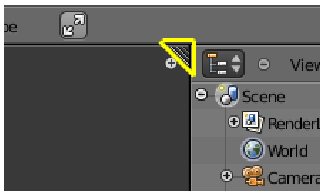
Arranging frames

Blender uses a novel screen-splitting approach to arrange window frames. The application window is always a rectangle on your desktop. Blender divides it up into a number of re-sizable window frames. A window frame contains the workspace for a particular type of window, like a 3D View window, or an Outliner. The idea is that you split up that big application window into any number of smaller (but still rectangular) non-overlapping window frames. That way, each window is always fully visible, and it is very easy to work in one window and hop over to work in another.

Maximizing a window

You can maximize a window frame to fill the whole application window with the View → Toggle Full Screen menu entry. To return to normal size, use again View → Toggle Full Screen. A quicker way to achieve this is to use ⇧ ShiftSpace, Ctrl↓ or Ctrl↑ to toggle between maximized and framed windows. NOTE: The window your mouse is currently hovering over is the one that will be maximized using the keyboard shortcuts.

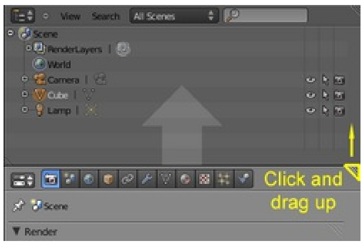
Splitting a window



In the upper right and lower left corners of a window are the window splitter widgets, and they look like a little ridged thumb grip. It both splits and combines window panes. When you hover over it, your cursor will change to a cross. LMB and drag it to the left to split the pane vertically, or downward to split it horizontally.

Joining two frames

In order to merge two window frames, they must be the same dimension in the direction you wish to merge. For example, if you want to combine two window frames that are side-by-side, they must be the same height. If the one on the left is not the same as the one on the right, you will not be able to combine them horizontally. This is so that the combined window space results in a rectangle. The same rule holds for joining two window frames that are stacked on top of one another; they must both have the same width. If the one above is split vertically, you must first merge those two, and then join the bottom one up to the upper one.



To merge the current window with the one above it (in the picture the properties window is being merged "over" the Outliner), hover the mouse pointer over the window splitter. When the pointer changes to a cross, LMB click and drag up to begin the process of combining. The upper window will get a little darker, overlaid with an arrow pointing up. This indicates that the lower (current) frame will "take over" that darkened frame space. Let go of the LMB to merge. If you want the reverse to occur, move your mouse cursor back into the original (lower) frame, and it will instead get the arrow overlay.

In the same way, windows may be merged left to right or vice versa.

If you press Esc before releasing the mouse, the operation will be aborted.

Changing window size

You can resize window frames by dragging their borders with LMB. Simply move your mouse cursor over the border between two frames until it changes to a double-headed arrow, and then click and drag.

Swapping contents

You can swap the contents between two frames with Ctrl LMB on one of the splitters of the initial frame, dragging towards the target frame, and releasing the mouse there. Those two frames don't need to be side by side, though they must be inside the same window.

Opening new windows

You may wish to have a new full window containing Blender frames. This can be useful, for instance, if you have multiple monitors and want them to show different information on the same instance of Blender.

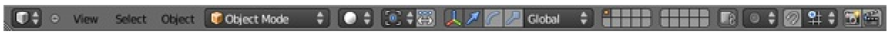
All you need to do is ⇧ Shift LMB on a frame splitter, and drag slightly. A new window pops up, with its maximize, minimize, close and other buttons (depending on your platform), containing a single frame with a duplicate of the initial window on which you performed the operation.

Once you have that new window, you can move it to the other monitor (or leave it in the current one); you can resize it (or keep it unchanged); you can also arrange its contents in the same way discussed so far (split and resize frames, and tune them as needed), and so on.

There is, though, another way to get an extra window: *File → User Preferences...* (or CtrlAltU) pops a new window also, with the *User Preferences* window in its only frame. You can then proceed the same way with this window.

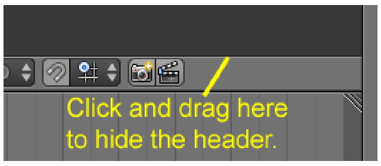
Window Headers

All windows have a header (the strip with a lighter gray background containing icon buttons). We will also refer to the header as the window *ToolBar*. The header may be at the top (as with the Properties Window) or the bottom (as with the 3D Window) of a window's area. The picture below shows the header of the 3D window:



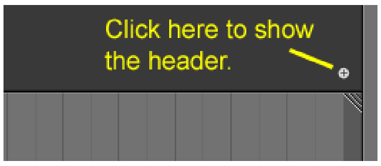
If you move the mouse over a window, its header changes to a slightly lighter shade of gray. This means that it is "focused". All hotkeys you press will now affect the contents of this window.

Hiding a header



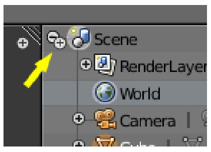
To hide a header, move your mouse over the thin line between a window and its header, until the pointer takes the form of an up/down arrow. Then click, hold and drag with LMB from the window over the header to hide the latter.

Showing a header



A hidden header leaves a little plus sign (see picture). By LMB this, the header will reappear.

Note 1: In the 3D window, there are up to two more of these little plus signs (to the top left and right of the window). Those will open panels with several tools, not a second header.



Note 2: In some windows, the mentioned plus sign can be hard to find, because it might look like a part of other icons. One example is the Outliner, in which there are other such plus signs, thus giving the one to get the header back good camouflage.

Header position

To move a header from top to bottom or the other way round, simply RMB on it and select the appropriate item from the popup menu. If the header is at the top, the item text will read "Flip to Bottom", and if the header is at the bottom the item text will read "Flip to Top".

Theme colors

Blender allows for most of its interface color settings to be changed to suit the needs of the user. If you find that the colors you see on screen do not match those mentioned in the Manual then it could be that your default theme has been altered. Creating a new theme or selecting/altering a pre-existing one can be done by selecting the User Preferences window and clicking on the Themes tab of the window.

Window type button

LMB clicking on the first icon at the left end of a header allows selection of one of the 16 different window types. Every window frame in Blender may contain any type of window. So if you want 3D views everywhere, just go ahead and change them all.

Menus and buttons

Most Window Headers, located immediately next to this first "Window Type" Menu button, exhibit a set of menus which can be hidden - again with a little minus sign. So if you cannot find a menu that was mentioned somewhere, try looking for a little plus sign (once again) next to the "Window Type" button. By clicking LMB on it, the menu will come back.

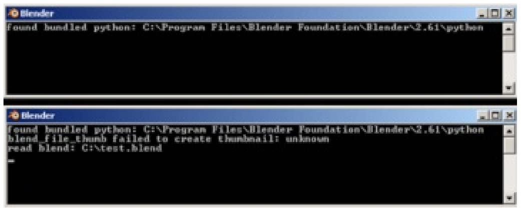
Menus allow you to directly access many features and commands, so just look through them to see what's there. All Menu entries show the relevant shortcut keys, if any.

Menus and buttons will change with Window Type and the selected object and mode. They only show the actions that can be performed.

The Console Window

The Console Window is an operating system text window that displays messages about Blender operations, status, and internal errors. If Blender crashes on you, the Console Window may be able to indicate the cause or error.

Windows XP/Vista/7



```
found bundled python: C:\Program Files\Blender Foundation\Blender\2.61\python
```

```
found bundled python: C:\Program Files\Blender Foundation\Blender\2.61\python
blend_file_thumb failed to create thumbnail: unknown
read blend: C:\test.blend
_
```

The Blender Console Window on Windows XP and subsequent messages.

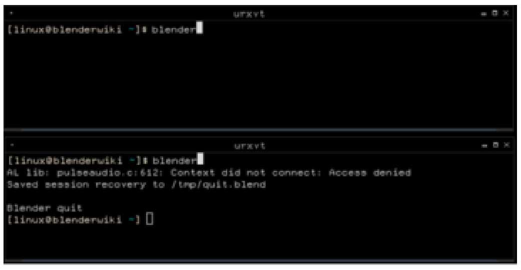
When Blender is started on a Windows operating system, the Console Window is first created as a separate window on the desktop. Assuming that the right start-up conditions are met, the main Blender window should also appear and the Console Window will then be toggled off. To display the console again, go to Window » Toggle System Console.

The screenshot on the right shows the Blender Console Window on Windows XP directly after starting Blender and then a short while later after opening a file along with the relevant messages.

Closing the Blender Console Window

The Blender Console Window must remain open while Blender is running. Closing the Console Window will also close Blender, and unsaved work would be lost. To turn off the console without closing Blender, toggle the console state to off via re-selecting Toggle System Console option from the drop-down menu Window » Toggle System Console. Note the Blender Console Window can look very similar to MS-DOS, so make sure that you are closing the correct window if an instance of MS-DOS is open.

Linux



```
[linux@blenderwiki ~]# blender
```

```
[linux@blenderwiki ~]# blender
AL lib: pulseaudio.c:612: Context did not connect: Access denied
Saved session recovery to /tmp/quit.blend

Blender quit
[linux@blenderwiki ~] 
```

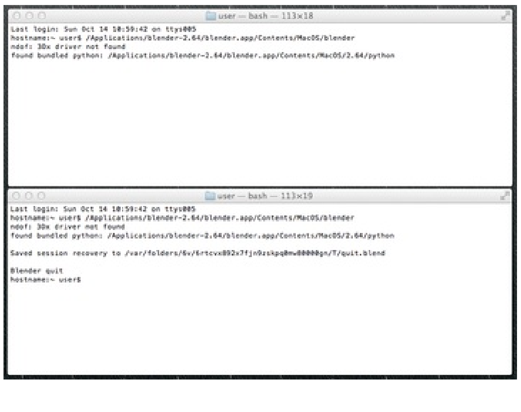
Starting Blender from a Linux console window and subsequent messages.

The Blender Console Window in Linux will generally only be visible on the Desktop if Blender is started from a Linux Terminal/Console Window as Blender uses the Console Window it is started from to display Console output.

Depending on your Desktop Environment setup, a Blender icon may appear on your desktop or an entry for Blender added to your menu after you install Blender. When you start Blender using a Desktop icon or menu entry rather than a Terminal window, the Blender Console Window text will most likely be hidden on the Terminal that your XWindows server was started from.

This screenshot shows Blender started from a Linux Terminal/Console Window and the resulting console text being printed to it. This example shows that when Blender was started it was unable to access a library related to the Pulseaudio sound server. When Blender closed, it saved the recovery file to */tmp/quit.blend*.

MacOS



```
Last login: Sun Oct 14 10:59:42 on ttys005
hostname:~ user$ /Applications/blender-2.64/blender.app/Contents/MacOS/blender
ndof: 3Dx driver not found
found bundled python: /Applications/blender-2.64/blender.app/Contents/MacOS/2.64/python
```

```
Last login: Sun Oct 14 10:59:42 on ttys005
hostname:~ user$ /Applications/blender-2.64/blender.app/Contents/MacOS/blender
ndof: 3Dx driver not found
found bundled python: /Applications/blender-2.64/blender.app/Contents/MacOS/2.64/python

Saved session recovery to /var/folders/6v/6rtcvx892x7fjn9zskpq8mw0000gn/T/quit.blend

Blender quit
hostname:~ user$
```

Starting Blender from a Mac OS X console window
and subsequent messages.

The process in MacOS is very similar to the one described for Linux. MacOS uses "files" with the .app extension called *applications*. These files are actually folders that appear as files in Finder. In order to run Blender you will have specify that path to the Blender executable inside this folder, to get all output printed to the terminal. You can start a terminal from Applications -> Utilities. The path to the executable in the .app folder is *./blender.app/Contents/MacOS/blender*.

If you have Blender installed in the Applications folder, the following command could be used, adapted to the particular Blender version: */Applications/blender-2.64/blender.app/Contents/MacOS/blender*

Console Window Status and Error Messages

The Blender Console Window can display many different types of Status and Error Messages. Some messages simply inform the user what Blender is doing, but have no real impact on Blender's ability to function. Other messages can indicate serious errors that will most likely prevent Blender carrying out a particular task and may even make Blender non-responsive or shut down completely. The Blender Console Window messages can also originate internally from within the Blender code or from external sources such as Python scripts.

Common messages

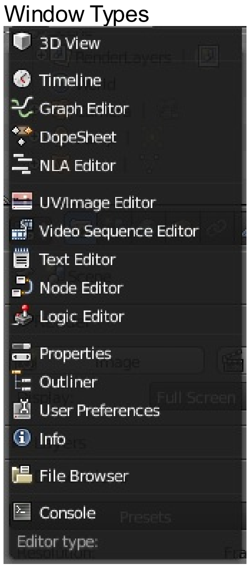
- found bundled python: (FOLDER)

  This message indicates that Blender was able to find the Python library for the Python interpreter embedded within Blender. If this folder is missing or unable to be found, it is likely that an error will occur, and this message will not appear.

- malloc returns nil()

  When Blender carries out operations that require extra memory (RAM), it calls a function called malloc (short for memory allocate) which tries to allocate a requested amount of memory for Blender. If this cannot be satisfied, malloc will return nil/null/0 to indicate that it failed to carry out the request. If this happens Blender will not be able to carry out the operation requested by the user. This will most likely result in Blender operating very slowly or shutting down. If you want to avoid running out of memory you can install more memory in your system, reduce the amount of detail in your Blender models, or shut down other programs and services which may be taking up memory that Blender could use.

Window Types



The Window Type selection menu.

The Blender interface is divided up into many rectangular Window Frames. Each Window Frame may contain different types of information, depending upon the Window Type.

Each Window Frame operates independently of the others, and you can have the same Window Type in many frames. For example, you may have several 3D windows open but each looking at the Scene from a different perspective. You can split and merge and resize Window Frames to suit whatever you are working on. You can also arrange some Window Frames to display without a Header to save screen space.

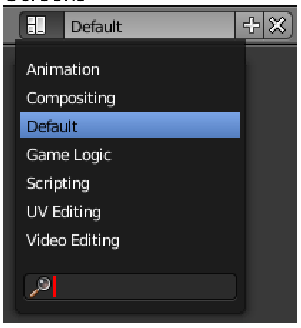
Read more about arranging frames »

Window Types are broken up by functionality:

- The 3D View- Show a graphical view of your scene.
- The Timeline - Control animation playback.
- The Graph Editor - Manage animation keys (and drivers) and inter/extrapolation of these.
- The Dope Sheet - Combine individual actions into action sequences.
- The NLA Editor - Manage non-linear animation action sequences.
- The Image/UV Editor - Edit images with advanced UV management tools.
- The Video Sequence Editor - Assemble video sequences into a film strip.
- The Text Editor - Keep notes and documentation about your project, and write Python scripts.
- The Node Editor - Use nodes for texturing materials and compositing.
- The Logic Editor - Edit game logic.
- The Properties Editor - Show several attributes of the currently selected object.
- The Outliner - Find and organize your objects.
- User Preferences - Customize Blender to your work style and computer.
- The Info Window - Provides information and options for managing files, windows and engines.
- The File Browser - Organize, load and save files (most times invoked automatically, when needed).
- The Console - Directly use python in Blender.

You can select the Window Type by clicking the Window Header's *leftmost* button. A pop-up menu displays showing the available Window Types.

Screens



Layout dropdown

Blender's flexibility with windows lets you create customized working environments for different tasks such as modeling, animating, and scripting. It is often useful to quickly switch between different environments within the same file.

To do each of these major creative steps, Blender has a set of pre-defined *screens*, that show you the types of windows you need to get the job done quickly and efficiently. *Screens* are essentially pre-defined window layouts. If you are having trouble finding a particular screen, you can use the search function at the bottom of the list (pictured right).

Default Screens available

Animation
: Making actors and other objects move about, change shape or color, etc.

Compositing
: Combining different parts of a scene (e.g. background, actors, special effects) and filter them (e.g. color correction).

Default
: The default layout used by Blender for new files. Useful for modeling new objects.

Game Logic
: Planning and programming of games within Blender.

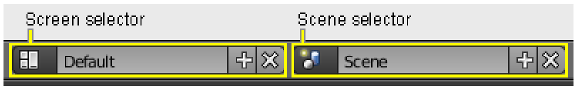
Scripting
: Documenting your work and/or writing custom scripts to automate Blender.

UV Editing
: Flattening a projection of an object mesh in 2D to control how a texture maps to the surface.

Video Editing
: Cutting and editing of animation sequences.

Blender sorts these screen layouts for you automatically in alphabetical and/or numerical order. The list is available in the Info Window header that is at the top of the layout for preset screens. This is often confused for a menu bar by those new to Blender; however it is simply a window showing only its header.

To change to the next alphabetically listed screen press Ctrl→; to change to the previous screen, press Ctrl←.



Screen and Scene selectors

By default, each screen layout 'remembers' the last scene it was used on. Selecting a different layout will switch to the layout **and** jump to that scene.

All changes to windows, as described in Window system and Window types, are saved within one screen. If you change your windows in one screen, other screens won't be affected.

Configuring your Screens

Adding a new Screen Type

Click on the "Add" button(⊞) and a new frame layout will be created based on your current layout.

You might want to give the new screen not only a *name* but also a *number* in front of it so that you can predictably scroll to it using the arrow keys. You can rename the layout by LMB in the field and typing a new name, or clicking again to position the cursor in the field to edit. For example you could use the name "6-MyScreen". See *Screen and Scene selectors* above.

Deleting a Screen

You can delete a screen by using the Delete datablock button (⊠). See *Screen and Scene selectors* above.

Rearranging a Screen

Use the window controls to move frame borders, split and consolidate windows. When you have a layout that you like, press CtrlU to update your User defaults. Be aware that all of the current scenes become part of those defaults, so consider customizing your layouts with only a single, simple scene.

The properties window has a special option: pressing RMB on its background will allow you to arrange its panels horizontally or vertically. Of the two, vertically-arranged panels have greater support.

Overriding Defaults

When you save a .blend file, the screen layouts are also saved in it. When you open a file, enabling the Load UI checkbox in the file browser indicates that Blender should use the file's screen layouts (overriding your defaults in the process). Leaving the Load UI checkbox disabled tells Blender to use the current layout.

Additional Layouts

As you become more experienced with Blender, consider adding some other screen layouts to suit your workflow as this will help increase your productivity. Some examples could include:

1-Model

Four 3D windows (top, front, side and perspective), Properties window for Editing

2-Lighting

3D windows for moving lights, UV/Image Window for displaying Render Result, Properties window for rendering and lamp properties and controls

3-Material

Properties window for Material settings, 3D window for selecting objects, Outliner, Library script (if used), Node Editor (if using Node based materials)

4-UV Layout

UV/Image Editor Window, 3D Window for seaming and unwrapping mesh

5-Painting

UV/Image Editor for texture painting image, 3D window for painting directly on object in UV Face Select mode, three mini-3D windows down the side that have background reference pictures set to full strength, Properties window

6-Animation

Graph Editor, 3D Window for posing armature, NLA Window

7-Node

Big Node Editor window for noodles, UV/Image window linked to Render Result

8-Sequence

Graph Editor, video sequence editor in Image Preview mode, video sequence editor in timeline mode, a Timeline window, and the good old Properties window.

9-Notes/Scripting

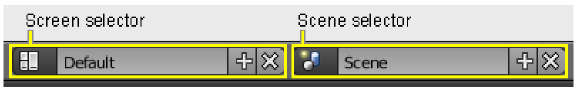
Outliner, Text Editor (Scripts) window

Reuse your Layouts
If you create a new window layout and would like to use it for future .blend files, simply save it as the User default by pressing CtrlU (don't forget: all screens and scenes themselves will be saved as default too).

Scenes

Scenes are a very useful tool for managing your projects. The Cube model in empty space you see when you open Blender for the first time is the default Scene. You can imagine Scenes to be similar to tabs in your web browser. For example, your web browser can have many tabs open at once. The tabs could be empty, showing identical views of the same web page, showing different views of the same page or show entirely different pages altogether. Blender's Scenes work in much the same way. You can have an empty Scene, a complete independent copy of your Scene or a new copy that links to your original Scene in a number of ways.

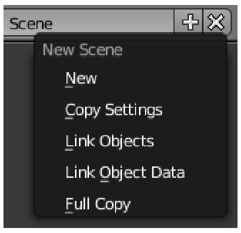
You can select and create scenes with the Scene selector in the Info window header (the bar at the top of most Blender layouts, see *Screen and Scene selectors*).



Screen and Scene selectors

Scene configuration

Adding a new Scene



Add Scene menu

You can add a new scene by clicking ⊞ in the Scene selector option. When you create a new scene, you can choose between five options to control its contents (*Add Scene menu*).

To choose between these options, you need to clearly understand the difference between "Objects" and "Object Data". Each Blender graphic element (Mesh, Lamp, Curve, *etc.*) is composed from two parts: an Object and Object Data (also known as ObData). The Object holds information about the position, rotation and size of a particular element. The ObData holds information about meshes, material lists and so on. ObData is common to every instance of that particular type of element. Each Object has a link to its associated ObData, and a single ObData may be shared by many Objects.

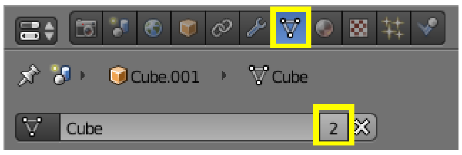
The five choices, therefore, determine just how much of this information will be *copied from* the currently selected Scene to the new one, and how much will be *shared* ("linked"):

New
: Creates an empty Scene. In the new Scene, the Render Settings are set to the default values.

Copy Settings
: Creates an empty Scene like the previous option but also copies the Render Settings from the original Scene into the new one.

Link Objects
: Is the shallowest form of copying available. This option creates the new Scene with the same contents as the currently selected Scene. However, instead of copying the Objects, the new Scene contains *links to* the Objects in the old Scene at the Object level. Therefore, changes in the *new*Scene will result in the same changes to the *original* Scene because the Objects used are *literally* the same Objects. The reverse is also true (changes in the *old* Scene will cause the same changes in the *new*Scene).

Link Object Data
: Creates new, duplicate copies of all of the Objects in the currently selected Scene, but each one of those duplicate Objects will have *links to* the ObData (meshes, materials and so on) of the corresponding Objects in the original Scene. This means that you can change the position, orientation and size of the Objects in the new Scene without affecting other Scenes, but any modifications to the ObData (meshes, materials *etc*) will also affect other Scenes. This is because a *single instance of* the "ObData" is now being shared by all of the Objects in all of the Scenes that link to it. If you want to make changes to an Object in the new Scene independently of the Objects in the other Scenes, you will have to manually make the object in the new Scene a "single-user" copy by LMB the number in the Object Data panel of the Properties Window. More information at the [Window Type](#) page. This has the effect of making a new independent copy of the ObData.



Full Copy
: Is the deepest form of copying available. Nothing is shared. This option creates a fully independent Scene with copies of the currently selected Scene's contents. Every Object in the original Scene is duplicated, and a duplicate, private copy of its ObData is made as well.

To better understand the way Blender works with data, read through [Blender's Library and Data System.](#)

A brief example

Consider a bar Scene in a film. You initially create the bar as a clean version, with everything unbroken and in its proper place. You then decide to create the action in a separate Scene. The action in your Scene will indicate which type of linking (if any) would suit your Scene best.

Link Objects
: Every object will be linked to the original Scene. If you correct the placement of a wall, it will move in every Scene that uses the bar as a setting.

Link Object Data
: Will be useful when the positions of Objects need to change, but their shape and material settings will remain constant. For example, chairs might stand on the floor in the "crowded bar" scene and up on the tables in the "we are closing" scene. Since the chairs don't change form, there is no need to waste memory on exact mesh-copies.

Full Copy
: A glass shattering on the floor will need its own copy because the mesh will change shape.

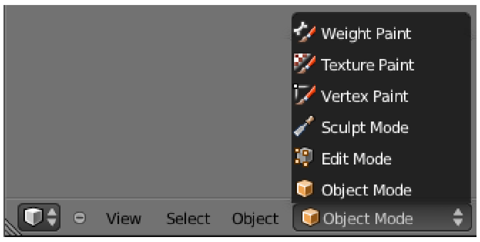
It is not possible to do all of the above in the same Scene, but it might help in understanding why to link different Objects in different ways.

Deleting a Scene

You can delete a scene by using the Delete datablock button (☒) from the Scene selector option (see *Screen and Scene selectors*).

Modes

Modes are a Blender-level object-oriented feature, which means that *the whole Blender application* is always in *one and only one mode*, and that the available modes vary depending on the selected active object's type – most of them only enable the default Object mode (like cameras, lamps, etc.). Each mode is designed to edit an aspect of the selected object. See the *Blender's Modes* table below for details.



Mode selection example (mesh object).

You set the current mode in the Mode drop-down list of 3D View header (see *Mode selection example (mesh object)*).

You can only select objects in Object mode. In all others, the current object selection is “locked” (except, to some extent, with an armature's Pose mode).

Modes might affect many things in Blender:

- They can modify the panels and/or controls available in some Buttons windows' contexts.
- They can modify the behavior of whole windows, like e.g. the UV/Image Editor window (and obviously, 3D Views!).
- They can modify the available header tools (menus and/or menu entries, as well as other controls…). For example, in the 3D View window, the Object menu in Object mode changes to a Mesh menu in Edit mode (with an active mesh object!), and a Paint menu in Vertex Paint mode…

Blender's Modes

Icon	Name	Shortcut	Remarks
	Object mode	*None*1	The default mode, available for all object types, as it is dedicated to Object datablock editing (i.e. position/rotation/size).
	Edit mode	⇆ Tab1	A mode available for all renderable object types, as it is dedicated to their “shape” ObData datablock editing (i.e. vertices/edges/faces for meshes, control points for curves/surfaces, etc.).
	Sculpt mode	*None*1	A mesh-only mode, that enables Blender's mesh 3D-sculpting tool.
	Vertex Paint mode	*None*1	A mesh-only mode, that allows you to set your mesh's vertices colors (i.e. to “paint” them).
	Texture Paint mode	*None*1	A mesh-only mode, that allows you to paint your mesh's texture directly on the model, in the 3D views.
	Weight Paint mode	Ctrl⇆ Tab2	A mesh-only mode, dedicated to vertex group weighting.
	Particle mode	*None*1	A mesh-only mode, dedicated to particle systems, useful with editable systems (hair).
	Pose mode	Ctrl⇆ Tab2	An armature-only mode, dedicated to armature posing.

Notes about modes shortcuts:

1. ⇆ Tab toggles Edit mode.
2. Ctrl⇆ Tab switches between the Weight Paint (meshes)/Pose (armatures) modes, and the other current one (by default, the Object mode). However, the same shortcut has other, internal meanings in some modes (e.g. in Sculpt mode, it is used to select the current brush)…

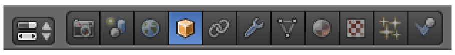
As you can see, using shortcuts to switch between modes can become quite tricky, especially with meshes…

We won't detail further more modes' usages here. Most of them are tackled in the modeling chapter, as they are mainly related to this topic. The Particle mode is discussed in the particle section, and the Pose and Edit modes for armatures, in the rigging one.

Note
If you are reading this manual and some button or menu option is referenced that does not appear on your screen, it may be that you are not in the proper mode for that option to be valid.

Contexts

The Properties (or Buttons) Window shows several Contexts, which can be chosen via the icon row in the header (see *Context button example*).

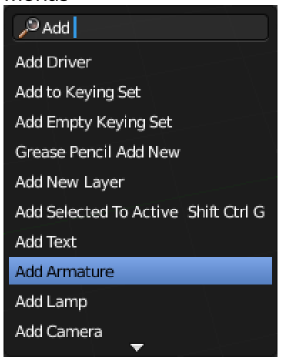

Context button example

The number and type of buttons changes with the selected Context so that only useful buttons show up. The order of these buttons follows a hierarchy which is detailed below:

- Render: Everything related to render output (dimensions, anti-aliasing, performance etc).
- Scene: Gravity in the scene, units and other general information.
- World: Environmental lighting, sky, mist, stars and Ambient Occlusion.
- Object: Transformations, display options, visibility settings (via layers) duplication settings and animation information (regarding Object position).
- Constraints: Used to control an Object's transform (position, rotation, scale), tracking and relationship properties.
- Modifiers: Operations that can non-destructively affect Objects by changing how they are rendered and displayed without altering their geometry (e.g. mirror and smoothing).
- Object Data: Contains all Object specific data (color of a lamp, focal length of a camera, vertex groups etc). The icon differs with the type of Object (the one shown here is for a mesh object).
- Materials: Information about a surface (color, specularity, transparency, etc).
- Textures: Used by materials to provide additional details (e.g. color, transparency, fake 3-dimensional depth).
- Particles: Add variable amounts of (usually small) objects such as lights or mesh Objects that can be manipulated by Force Fields and other settings.
- Physics: Properties relating to Cloth, Force Fields, Collision, Fluid and Smoke Simulation.

The Buttons in each context are grouped into Panels.

Menus



The Space-menu

Blender contains many menus, each of which is accessible from either a window's header or directly at the mouse's location using HotKeys or by clicking RMB on a window border, a button or elsewhere on the screen. A context sensitive menu will be displayed if there is one available for that interface element.

Additionally, a menu with access to all Blender commands is available by pressing Space (shown in the picture). Simply start typing the name of the command you need and let the search function of the menu do the rest. When the list is sufficiently narrowed, LMB on the desired command or highlight it with ↓ and ↑ and select with Return.

If you miss the old tool box menu from version 2.4x, you can add something similar with the 3D View: Dynamic Spacebar Menu Add-On which can be installed from the Add-Ons tab of the Preferences window.

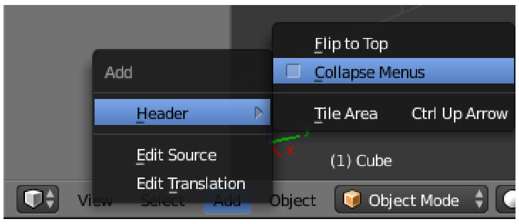
Read more about installing Add-Ons »

Some menus are context sensitive in that they are only available under certain situations. For example, the specials menu (W hotkey) is only available in a 3D window while Edit Mode is active.

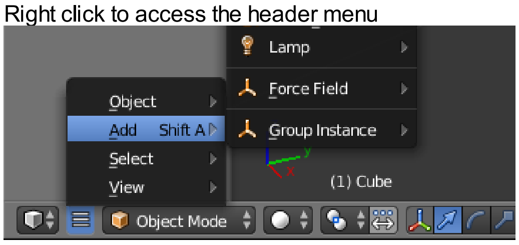
While you are using Blender, be aware of what mode is activated and what type of object is selected. This helps in knowing what hotkeys work at what times.

Collapsing Menus

Sometimes its helpful to gain some extra horizontal space in the header by collapsing menus, this can be accessed from the header context menu, simply right click on the header and enable set it to collapsed.



Right click to access the header menu



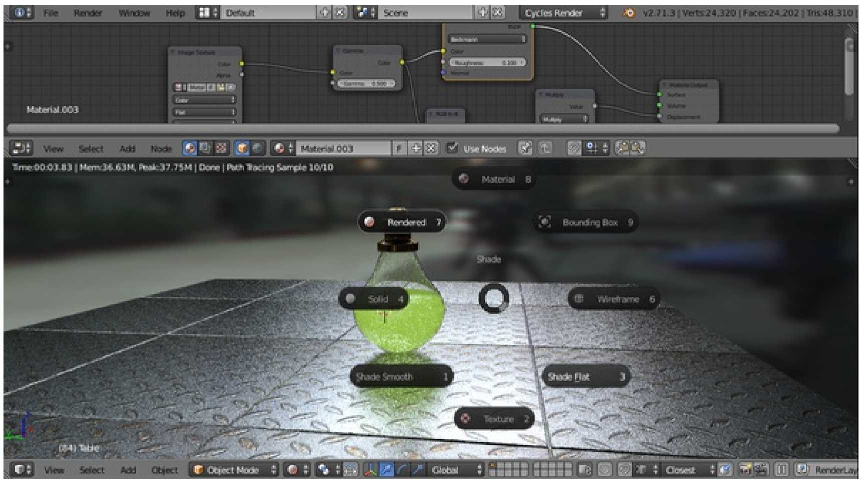
Access the menu from the collapsed icon

Menus on a Mac

Because Blender doesn't use the standard OS menu system, if you are using a Mac, you likely have a redundant menubar at the top. To remove it see this post on Macworld, but beware that it is somewhat complex. As an alternative: simply make Blender full screen with the last button in the info window header (most times at the top of the screen layout).

Pie menus

A pie menu is a menu whose items are spread radially around the mouse.



A pie menu in action. The currently active option, "Rendered", is highlighted by a bright outline, while a disc widget indicates the direction and a slight highlight indicates the pie menu item that will be selected on confirmation, "Shade Flat"

Interaction

To spawn a pie menu, a user needs to simply press the key that will spawn the menu.

- Releasing the key without moving the mouse will keep the menu open and the user can then move the mouse pointer towards the direction of a pie menu item and select it by clicking. (Also called click-style interaction)
- Releasing the key after moving the mouse towards a pie menu item will cause the menu to close and the selected menu item to activate. (Also called drag-style interaction)

An open disc widget at the center of the pie menu can help users discern the current direction of the pie menu. The selected item is also highlighted.
A pie menu will only have a valid direction for item selection if the mouse is touching or extending beyond the disc widget at the center of the menu. The radius of the widget is controllable by the threshold option in the user preferences.
On some, but not all pie menus, the currently active option is also highlighted by a bright outline (see picture).

Recentering
The window system tries to keep the pie menu within the window borders. This will naturally change the initial position of the pie menu if users try to spawn one close to the window borders. As a result of this, the initial orientation won't be neutral, but rather pointing to the item closest to the edge. To counter this, there is a recenter timeout option. If users increase this timeout, then pie menus will use the initial mouse position for that duration while the pie direction is calculated. This allows for fast dragged selections.

Positioning
Positions of items on a pie menu are meant to stay the same to preserve muscle memory. This means that if some options are missing due to context, other items won't move to fill in the blanks. Example: Meshes do not have a pose mode so ⇆ Tab pie menu will not show this option. However, Object and Edit mode options stay on the same direction. This is also important since adding new items in any existing menus should not change the old positions.

Key interaction
Pie menu items support key accelerators. The key accelerator is the letter that is underlined on each menu item. Also numbers can be used to select the items. Similarly to positioning, numbers are not meant to change if items are added or are missing due to different context so each number is bound to a specific direction. The numbering scheme corresponds to direction of numbers on a numberpad, taking number 5 as the center.

New Menus

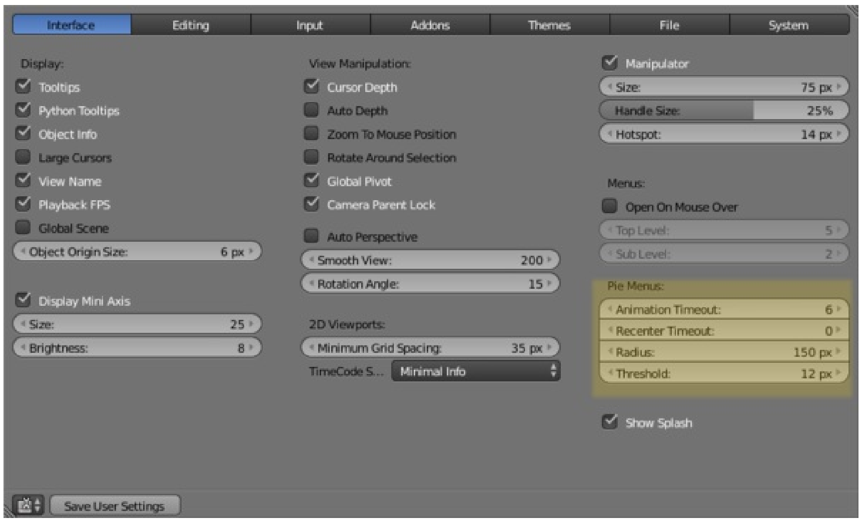
Pie menus will not be included in default blender. Rather, they can be activated through add-ons. Blender, in this first iteration, will include a minimal add-on that will ship the following pie menus:

- ⇆ Tab : Interaction Mode
- Z : Shade + solid vs smooth shading
- Q : View directions + perspective/ortho and camera
- , : Snapping
- . : Pivot
- CtrlSpace : Manipulator

To activate those menus, go to the Add-Ons tab in user preferences and enable the "Pie Menus Official" Add-On under the "User Interface" category.
Pie menus will completely override the operators that were previously assigned on those keys. There is, however, a sticky operator under development that will allow using both operators and pies on the same key

Options

There are a few options available in user preferences to tweak the operation of pie menus:



Display:
- [✓] Tooltips
- [✓] Python Tooltips
- [✓] Object Info
- [ ] Large Cursors
- [✓] View Name
- [✓] Playback FPS
- [ ] Global Scene
- Object Origin Size: 6 px
- [✓] Display Mini Axis
- Size: 25
- Brightness: 8

View Manipulation:
- [✓] Cursor Depth
- [ ] Auto Depth
- [ ] Zoom To Mouse Position
- [ ] Rotate Around Selection
- [✓] Global Pivot
- [✓] Camera Parent Lock
- [ ] Auto Perspective
- Smooth View: 200
- Rotation Angle: 15

2D Viewports:
- Minimum Grid Spacing: 35 px
- TimeCode S...: Minimal Info

- [✓] Manipulator
- Size: 75 px
- Handle Size: 25%
- Hotspot: 14 px

Menus:
- [ ] Open On Mouse Over
- Top Level: 5
- Sub Level: 2

Pie Menus:
- Animation Timeout: 6
- Recenter Timeout: 0
- Radius: 150 px
- Threshold: 12 px

- [✓] Show Splash

User preferences.

Animation Timeout This number is the time, in 1/100s of a second that the pie menu opening animation takes to finish.

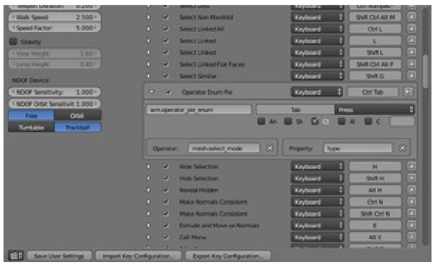
Recenter Timeout If a pie menu is close to the screen borders, the system will attempt to recenter a pie menu so that it is always visible. This means that the direction the user drags towards will be different from what was initially intended. However, the direction that is calculated before the recenter timeout has passed will always use the initial position that the menu was spawned from, so quick gestures are still possible, even when a menu is recentered.

Radius The distance, in pixels of the menu items from the center of the pie menu

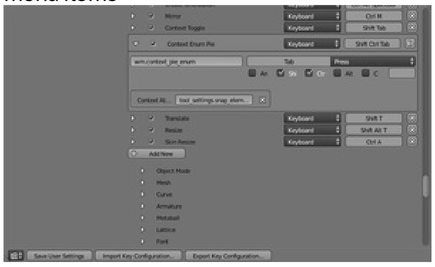
Threshold A valid selection can only be determined only after the mouse has at least this distance from the center of the pie menu. This helps to minimize accidentally selecting unwanted items.

Adding Pie Menus In Blender

There are a lot of ways to add a pie menu. For users, maybe the easiest way is to change a keymap to spawn a pie menu item for an operator or context enum. Each case has dedicated operators for that. wm.operator_pie_enum will spawn a pie based on an operator's enum property, while wm.context_pie_enum will spawn a pie to select between enum values of a context property. Below are two examples of such tweaks:



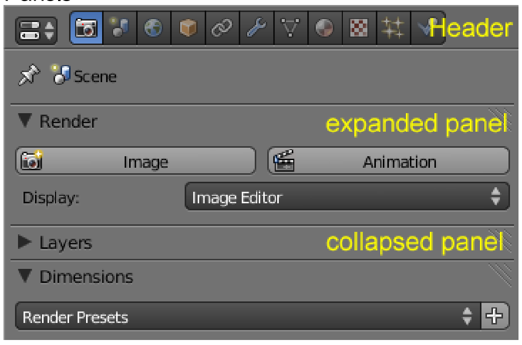
Operator Enum pie. This will spawn a pie menu on the edit mode selection operator using the "type" enum property on the pie menu items



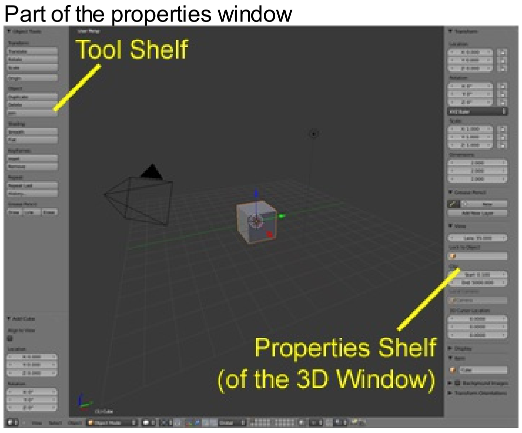
Context Enum pie. This will spawn a pie menu on snapping mode used in 3D viewport

Pie menus can also be coded in python, for more details Check: http://wiki.blender.org/index.php/Dev:Source/UI/Pie_Menus for more details

Panels



Part of the properties window



Shelves in a 3D Window

Panels generally appear in the Properties Window (Buttons Window in Version 2.4x), which can be found on the right hand side of the default screen layout (see *Part of the Properties window*).

Panels are also found on the Tool shelf and the Properties shelf which are toggleable parts of a 3D Window. To display the Tool shelf, use View » Tool or press T. To display the Properties shelf, use View » Properties or press N. See *Shelves in a 3D Window*.

The Properties Window includes the header which allows you to choose from several Contexts. Each Context will have a different number and type of Panels. For example, the Render Context will have panels that allow you to alter the dimensions and anti-aliasing of the render output, while the Materials Context will have panels that allow you to set color, transparency, texture, etc.

Panels in the Properties Window can be aligned vertically or horizontally by RMB on the Properties Window and choosing the desired option from the menu. Note that the Panels in the Properties Window are optimized for vertical alignment. Horizontal alignment may be cumbersome to work with.

The placement and view of panels can also be altered to your preference. For example, panels can be:

- moved around the window (or shelf) by LMB clicking, holding and dragging the widget in the upper right corner (this resembles the frame splitter widget and has three lines in a triangle formation).
- scrolled up and down by using Wheel
- zoomed in and out by holding Ctrl MMB and moving the mouse right and left.
- collapsed/expanded by LMB clicking the solid black triangle on the left side of their header.

For further details about each panel, see the Panels reference section, or find the appropriate section in the manual.

Buttons and Controls

Buttons and other controls can be found in almost every Window of the Blender interface. The different types of controls are described below.

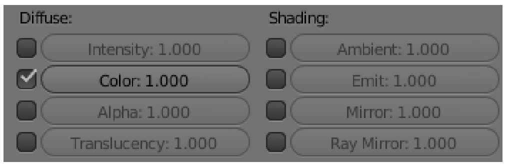
Operation Buttons

Operation button

These are buttons that perform an operation when clicked with LMB. They can be identified by their gray color in the default Blender scheme.

Pressing CtrlC over these buttons copies their python command into the clipboard which can be used in the python console or in the text editor when writing scripts.

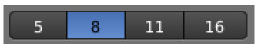
Toggle Buttons



Toggle buttons

Toggle buttons consist of tick boxes. Clicking this type of button will toggle a state but will not perform any operation. In some cases the button is attached to a number button to control the influence of the property.

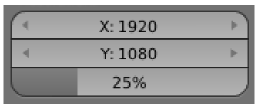
Radio Buttons



Radio buttons

Radio buttons are used to choose from a small selection of "mutually exclusive" options.

Number Buttons



Number buttons

Number buttons can be identified by their labels, which in most cases contains the name and a colon followed by a number. Number buttons are handled in several ways:

1. To change the value in steps, click LMB on the small triangles on the sides of the button.
2. To change the value in a wider range, hold down LMB and drag the mouse to the left or right. If you hold Ctrl after holding down LMB, the value is changed in discrete steps; if you hold ⇧ Shift instead, you'll have finer control over the values.
3. ↵ Enter or LMB lets you enter the value by hand.

When entering values by hand, pressing ↖ Home or ⇥ End will move the cursor to the beginning or the end of the range. Pressing Esc will cancel editing. You can copy the value of a button by hovering over it and pressing CtrlC. Similarly you can paste a copied value with CtrlV.

Expressions

You can also enter expressions such as `3*2` instead of `6`. or `5/10+3`. Even constants like `pi` (3.142) or functions like `sqrt(2)` (square root of 2) may be used.

These expressions are evaluated by python; for all available math expressions see: math module reference

Units

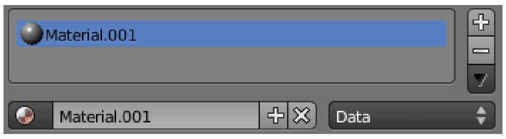
As well as expressions, you can mix units with numbers; for this to work, units need to be set in the scene settings (Metric or Imperial).

Examples of valid units include:

- `1cm`
- `1m 3mm`
- `1m, 3mm`
- `2ft`
- `3ft/0.5km`
- `2.2mm + 5' / 3" - 2yards`

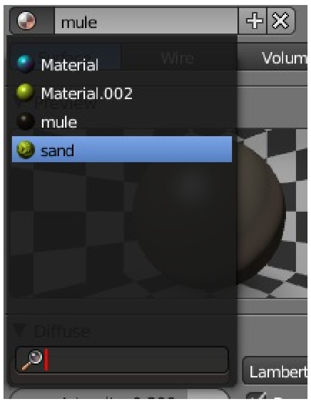
Note that the commas are optional. Also notice howyou can mix between metric and imperial even though the display can only *showone at a time.*

Menu Buttons



Datablock link buttons

Use the Menu buttons to work with items on dynamically created lists. Menu buttons are principally used to link DataBlocks to each other. DataBlocks are items like Meshes, Objects, Materials, Textures, and so on. Linking a Material to an Object will assign that material to the selected Objects.



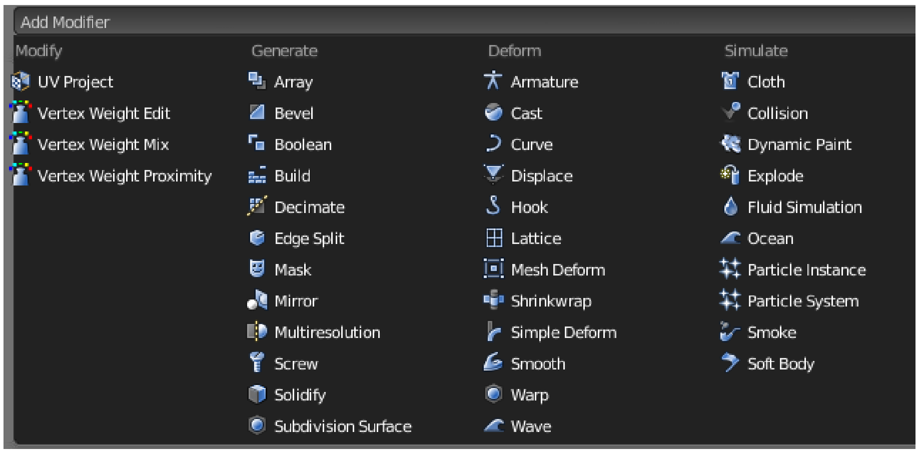
Datablock link menu with search

1. The first button (with an icon of the DataBlock type) opens a menu that lets you select the DataBlock to link by clicking LMB on the requested item. This list has a search box at the bottom.
2. The second button displays the name of the linked DataBlock and lets you edit it after clicking LMB.
3. The "+" button duplicates the current DataBlock and applies it.
4. The "X" button clears the link.

Sometimes there is a list of applied DataBlocks (such as a list of materials used on the object). See *DataBlock link buttons* above.

1. To select a datablock, click LMB on it.
2. To add a new section (e.g. material, or particle system), click LMB on the "+" button to the right of the list.
3. To remove a section, click LMB on the "-" to the right of the list.

Another type of a Menu button block will show a static list with a range of options. For example, the Add Modifier button will produce a menu with all of the available modifiers.



Modifier options

Unlinked objects

Unlinked data is ***not* lost until you quit Blender**. This is a powerful Undo feature. If you delete an object the material assigned to it becomes unlinked, but is still there! You just have to re-link it to another object or supply it with a "Fake User" (i.e. by clicking that option in the corresponding DataBlock in the datablock-view of the Outliner).

Read more about Fake User »

Color Selector Controls

In Blender, you can choose from **4** types of color pickers; the options are
Circle (Default), Square (HS + V) , Square (SV + H) and Square (HV + S)

For more information about how to select the type of color picker, please go to the System preferences page.

All of the Color picker types have the common RGB, HSV and Hex options to show values.
Optionally, depending on the operation, another slider for Alpha control is added at the bottom of the color picker.

Blender uses Floating point values to express colors for RGB and HSV values.
The Hex values are expressed in the same way HTML colors are expressed.

Note that Blender corrects Gamma by default; for more information about how to disable Gamma correction in Blender, please go to the *Color Management and Exposure* page.

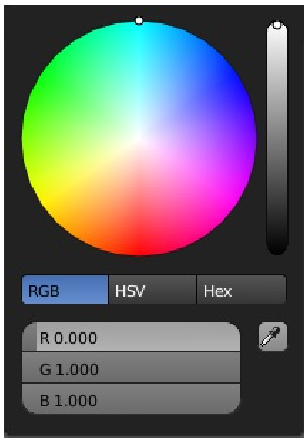


Fig. 2 - Color Picker - Circle

Circle (Default)
A full gamut of colors ranging from center to the borders is always shown; center is a mix of the colors.
Brightness is adjusted with the right bar, from top to bottom.
For operations that are capable of using Alpha, another slider is added at the bottom of the color picker.
See Fig. 2 - Color Picker - Circle

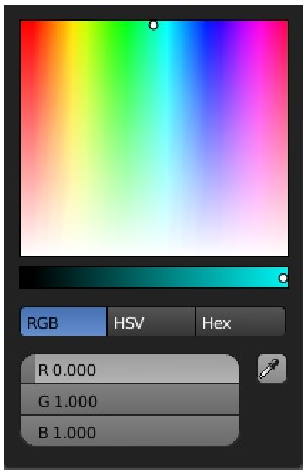


Fig. 3 - Color Picker
Square (HS + V)

Square (HS + V)
Hue, Saturation plus Value → A full gamut of colors is always shown.
Brightness is subtracted from the base color chosen on the square of the color picker moving the slider to the left.
For operations that are capable of using Alpha, another slider is added at the bottom of the color picker.
See Fig. 3 - Color Picker - Square (HS + V)

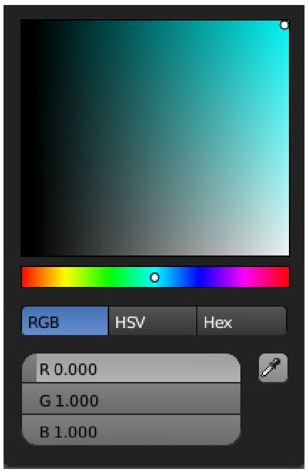


Fig. 4 - Color Picker
Square (SV + H)

Square (SV + H)
: Saturation, Value plus Hue → A full Gamut of colors is always shown at the bar in the middle of the color picker. Colors are adjusted using the a range of brightness of the base color chosen at the color bar in the middle of the picker. For operations that are capable of using Alpha, another slider is added at the bottom of the color picker. See Fig. 4 - Color Picker - Square (SV + H)

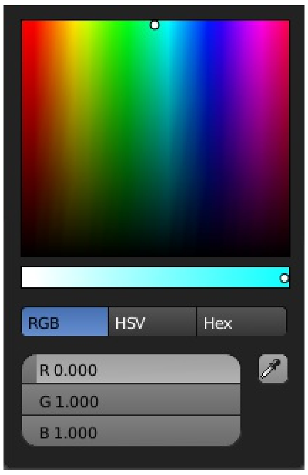


Fig. 5 - Color Picker
Square (HV + S)

Square (HV + S)
: Hue, Value and Saturation → A full gamut of colors is always shown at the square of the color picker. Brightness is added to the base color chosen on the square of the color picker moving the slider to the left. For operations that are capable of using Alpha, another slider is added at the bottom of the color picker. See Fig. 5 - Color Picker - Square (HV + S)

- Use Mouse wheel to change overall brightness.
- Color sliders don't have a default value; the last value before any changes is used instead.

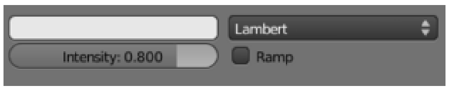
Eye Dropper

The eye dropper allows you to sample a color from anywhere in the Blender window. The Eye Dropper can be accessed from any color picker or by pressing E with the mouse hovering over the color property.

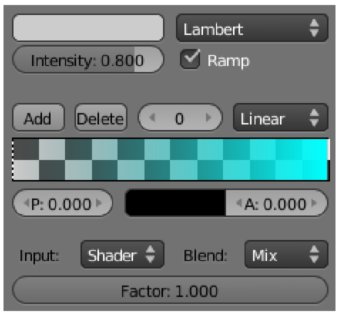
LMB and dragging the eyedropper will mix the colors you drag over which can help when sampling noisy imagery. Spacebar resets and starts mixing the colors again.

Cascade Buttons

Occasionally, some buttons actually reveal additional buttons. For example, the Ramps panel has a Cascade button called Ramp that reveals additional buttons dealing with colorbanding. See *Colorband before* and *Colorband after*.



Colorband before



Colorband after

Color Ramps
: Color Ramps enables the user to specify a range of colors based on color stops.
Color stops are similar to a mark indicating where the exact chosen color should be.
The interval from each of the color stops added to the ramp is a result of the color interpolation and chosen interpolation method. The available options for Color Ramps are:

Add (Button)
: Clicking on this button will add a stop to your custom weight paint map.
The stops are added from the last selected stop to the next one, from left to right and they will be placed in the middle of both stops.

Delete (Button)
: Deletes the selected color stop from the list.

'F' (Button)
: Flips the color band, inverting the values of the custom weight paint range.

Numeric Field
: Whenever the user adds a color stop to the custom weight paint range, the color stop will receive an index.
This field shows the indexes added (clicking in the arrows until the counter stops), and allows the user to select the color stop from the list. The selected color stop will be shown with a dashed line.

Interpolation Options
: Enables the user to choose from **4** types of calculations for the color interpolation for each color stop.
Available options are:

B-Spline
: Uses a B-Spline Interpolation for the color stops.

Cardinal
: Uses a Cardinal Interpolation for the color stops.

Linear
: Uses a Linear Interpolation for the color stops.

Ease
: Uses a Ease Interpolation for the color stops.

Constant
: Uses a Constant Interpolation for the color stops.

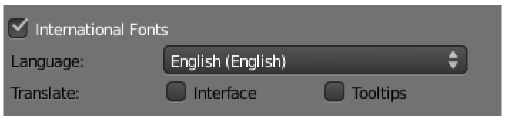
Position
: This slider controls the positioning of the selected color stop in the range.

Color Bar
: Opens a color Picker for the user to specify color and Alpha for the selected color stop.
When a color is using Alpha, the Color Bar is then divided in two, with the left side showing the base color and the right side showing the color with the alpha value.

Blender Internationalization

From version 2.60, Blender supports international fonts and a range of language options for the Interface and Tooltips. To enable it, open the User Preferences window, System tab, and toggle the International Fonts option in the bottom right-hand corner.

This displays three new settings:

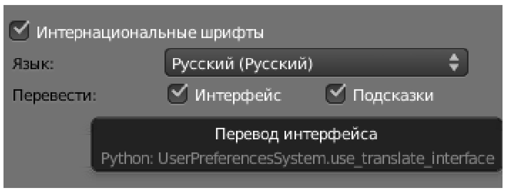


Enabling international fonts in the User Preferences window.

Language
: drop-down menu where you can select your preferred language.

Interface
: to translate the User Interface itself (e.g. controls and menus).

Tooltips
: to translate tooltips.



Blender with the Russian language enabled for the Interface and Tooltips.

Blender refreshes the screen after selecting Interface or Tooltips (or both) to show the new language. Note that some language translations are not yet complete. The progress of each translation is indicated in the drop-down menu.

Tip

Since the majority of tutorials are done with an English User Interface, it may be useful to keep the User Interface in English and only translate the tooltips, but it's important to note that the Blender User Interface translation, user manuals translation and development, together with community work, could develop a strong Blender user base, promoting Blender and localized Blender jobs!

Quick Rendering

What is rendering?

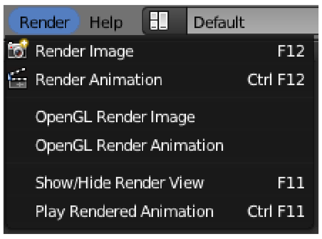
Rendering is the process of creating a 2D image. Blender creates this image by taking into account your model and all of your materials, textures, lighting and compositing.

- There are two main types of rendering engines built inside Blender, one for *Full render*, and other for *OpenGL render*. This page shows you basic information about rendering Images. For a deeper understanding about the *Full Render* Engine built inside Blender, called Blender Internal, consult the section about Rendering with Blender Internal.
- There is also a section in this wiki manual dedicated to the new Cycles Render Engine, built into Blender since Version 2.61.

Rendering an image using *Full Render* - Blender Internal

Mode: All modes

Hotkey: F12


Render
Help
Default
Render Image F12
Render Animation Ctrl F12
OpenGL Render Image
OpenGL Render Animation
Show/Hide Render View F11
Play Rendered Animation Ctrl F11
Header of the Info Window

To start a *Full render* using Blender Internal you can use any of the following options:

- Press F12
- Go to Properties Window » Render context » Render panel and press the Image button
- Go to Render » Render Image from the header of the Info Window (see: *Header of the Info Window*)
- Using Blender Search: press Space, type Render and click on Render.

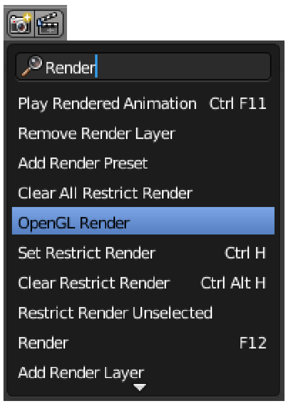
To abort or quit the render, press Esc.

Rendering an image using *OpenGL Render*

Mode: All modes

Hotkey: Undefined -You can add one for your Keymap »

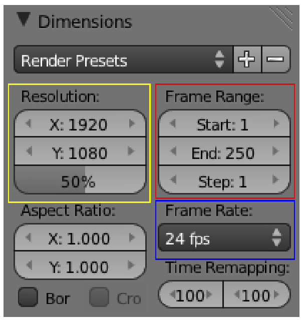
To start an *OpenGL render* you can use any of the following options:


Render
Play Rendered Animation Ctrl F11
Remove Render Layer
Add Render Preset
Clear All Restrict Render
OpenGL Render
Set Restrict Render Ctrl H
Clear Restrict Render Ctrl Alt H
Restrict Render Unselected
Render F12
Add Render Layer
Search functionality

- Click on *OpenGL Render Active Viewport*, in the header of the 3D Window, using the small button showing a *Camera* (together with a small image showing a *slate*) in the header of the 3D View
- Go to Render » OpenGL Render Image from the header of the Info Window (see: *Header of the Info Window* Image)
- Using Blender Search: press Space, type *Render* and click on OpenGL Render.

To abort or quit the render, press Esc.

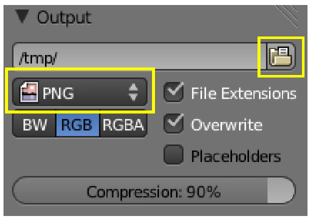
Adjusting the resolution



Dimensions panel

The Dimensions panel of the Render context allows you to change the resolution. The default installation of Blender is set initially to **50%** of **1920 x 1080**, resulting in a **960** x **540** Image. (Highlighted in yellow, in Dimensions Panel Image.) Higher resolutions and high percentage scales will show more detail, but will also take longer to render.

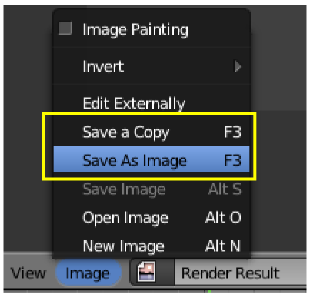
Output format and output file



Output panel

You can also choose an output format and the output location for your rendered image or animation. By default they are saved in a temporary folder (/tmp), using an absolute path. You can set up your file paths using instructions in the File setup chapter; however you can change this to a different folder by clicking the folder icon in the Output panel. You can also choose the type of image or movie format for your work from the Menu Button.

Saving your image



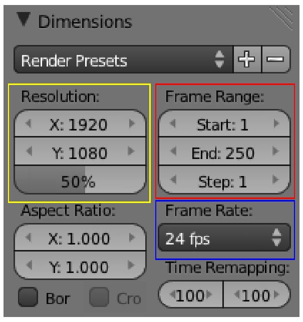
Save as dialog

Blender does not save your image automatically. To save your image, you can either press F3 or click Save As Image from the Image menu of the UV/Image editor window's header. This action will open the Blender Internal File Browser, and then you can search for folders to place your Render.

Rendering an animation using *Full Render* - Blender Internal

Mode: All modes

Hotkey: CtrlF12



Dimensions panel

Rendering an animation is simple; the Frame Range (Highlighted in red, in Dimensions Panel Image) in the Output Panel is used to define the **number of frames** your animation will render. The **time** is defined by the *Frames Per Second*, defined in the Frame Rate (Highlighted in blue, in Dimensions Panel Image) drop-down list. The default is set to **24 FPS** and **250** frames.

A quick example to understand those numbers:

- The Panel shows that the animation will start at frame **1** and end at frame **250**, and the FPS setting is set to **24**, so, the standard Blender installation will give you approximately **10** (ten) seconds of animation (250 / 24 = 10.41 sec).

To render an animation using *Full Render* with the Blender Internal Engine, you can use any of the following options:

- Press CtrlF12
- Go to Properties Window » Render context » Render panel and press the Animation button or
- Go to Render » Render animation from the header of the Info Window (see: *Header of the Info Window* Image)

To abort or quit rendering the animation, press Esc.

Rendering an animation using *OpenGL Render*

Mode: All modes

Hotkey: Undefined -You can add one for your Keymap »

To Render an animation using *OpenGL Render*, you can use any of the following options:

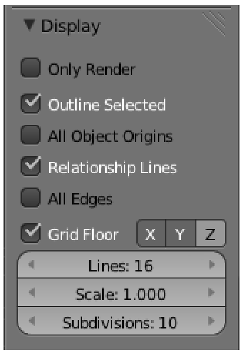
- Click on the small button showing a *slate* (together with a small image showing a *camera*) in the header of the 3D View
- Go to Render » OpenGL Render animation from the header of the Info Window (see: *Header of the Info Window* Image)

To abort or quit rendering the animation, press Esc.

Showing Only Rendered Objects

Mode: All modes

Hotkey: Undefined - You can add one for your Keymap »



Transform Panel - Display Tab.

At render time (either Full or OpenGL), there are some Objects in the scene that won't be rendered, either because of their type (Bones, Empties, Cameras, etc.), because they are void or have no visible geometry (Mesh without any vertex, curves not extruded, etc.), or simply because they are set as not renderable.

Blender has an option to only show Objects in the Scene that will be rendered.

To access this option, put your Mouse in a 3D View (focusing on it), use shortcut N or click in the **+** sign in the upper right side, to show the Transform Panel. Rolling through the options, you will find the Display tab, whose options are for controlling how Objects are displayed in the 3D View.

Just enable the Only Render option - now, only Objects that will be rendered will be shown (see Fig: Transform Panel - Display Tab). This option also works when generating Images using OpenGL Render. Note that all of the other options for selective displaying will be disabled.

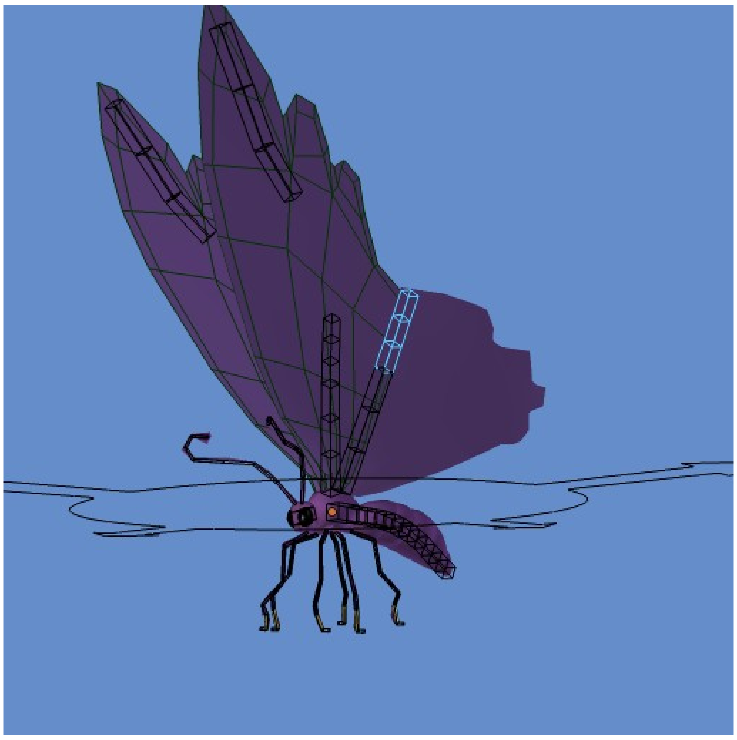
The purposes of OpenGL Rendering

OpenGL rendering allows you to quickly inspect your animatic (for things like object movements, alternate angles, etc.), by giving you a draft quality rendering of the current viewport.

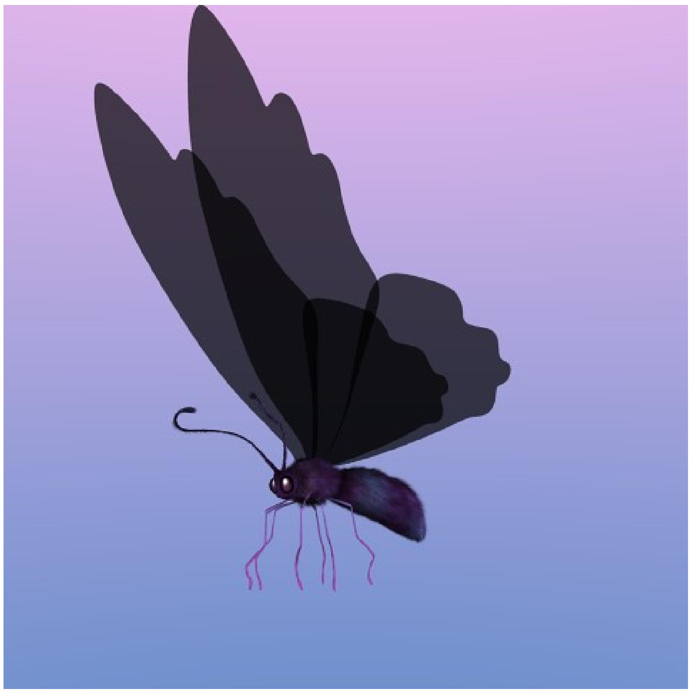
Because it is only rendered using OpenGL, it is much faster to generate, even if it only looks as good as what you see in the 3D viewport.

This allows you to preview your animation with fluid playback, which you would otherwise not be able to do in real time due to scene complexity (i.e., pressing AltA results in too low of a *Frames Per Second* to get a good feel for the animation).

This is an example of an OpenGL rendered image:



And then here is the *Full Render* using Blender Internal render engine:



You can use OpenGL to render both images and animations, and change dimensions using the same instructions explained above.
As with a normal render, you can abort it with Esc.

Recovering from mistakes or problems

Blender provides a number of ways for the user to recover from mistakes, and reduce the chance of losing their work in the event of operation errors, computer failures, or power outages. There are two ways for you to recover from mistakes or problems:

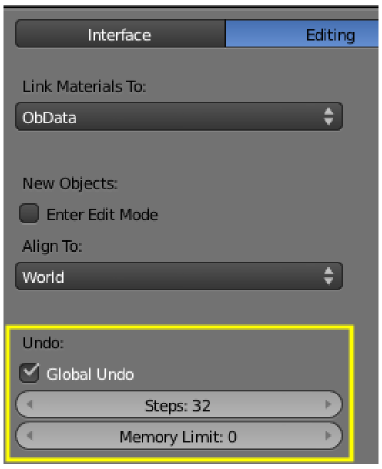
At the User Level (Relating to Actions)

- For your actions, there are options like Undo, Redo and an Undo History, used to roll back from mistakes under normal operation, or return back to a specific action.
- Blender also has new features like Repeat and Repeat History, and the new Redo Last which you can use in conjunction with the options listed.

At the System Level (Relating to Files)

- There are options to save your files like Auto Save that saves your file automatically over time, and Save on Quit, which saves your Blender file automatically when you exit Blender. Note: In addition to these functions being enabled by default, the Save on Quit functionality cannot be disabled.

Options for Actions (User Level)



Undo options

The commands listed below will let you roll back an accidental action, redo your last action, or let you choose to recover to a specific point, by picking from a list of recent actions recorded by Blender. Two new features that were added to the Blender 2.5x series are the Repeat and Repeat History commands.

To enable or disable Undo, go to the User Preferences window and click on the Editing tab. In this section you can set:

Global Undo
: This enables Blender to save actions done when you are **not** in Edit Mode. For example, duplicating Objects, changing panel settings or switching between modes. The default Blender Installation comes with the option *Global Undo* enabled.

Steps
: This numeric field indicates how many steps or actions to save. The default value of **32** will allow you to Undo the last thirty-two actions that you performed. You can change this numeric field to the maximum of **64**.

Memory Limit
: This numeric field allows you to define the maximum amount of memory in Megabytes that the Undo system is allowed to use. The default value of **0** indicates no limit.

Undo

Mode: All modes

Hotkey: CtrlZ

Like most programs, if you want to undo your last action, just press CtrlZ

Redo

Mode: All modes

Hotkey: ⇧ ShiftCtrlZ

To roll back your Undo action, press ⇧ ShiftCtrlZ

Redo Last

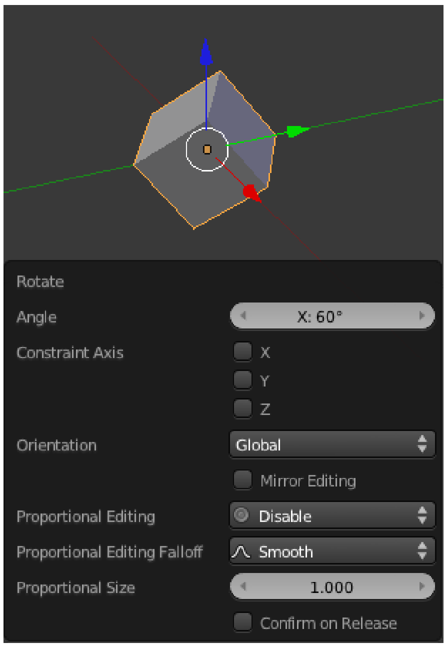
Mode: All modes

Hotkey: F6

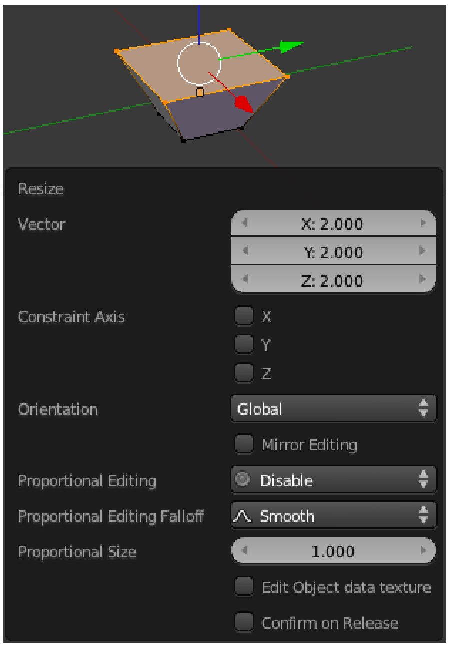
Redo Last (New feature) is short for Redo(ing your) Last (Action). Hitting F6 after an action will present you a context-sensitive Pop-Up Window based on your last action taken and the Mode and Window in which Blender is being used.

For example, if your last action was a rotation in Object Mode, the Window will show you the last value changed for the angle (see Fig:Redo Last - Rotation), where you can change your action back completely by typing **0** (zero) in the numeric field. There are other useful options, based on your action context, and you can not only Undo actions, but change them completely using the available options.

If you are in Edit Mode, the Window will also change its contents based on your last action taken. In our second example (at the right), the last action taken was a Vertex Move; we did a Scale on a Face, and, as you can see, the contents of the Pop-Up Window are different, because of your context (Edit Mode). (See Fig:Redo Last - Scale)



Rotate
Angle: X: 60°
Constraint Axis: X, Y, Z
Orientation: Global
Mirror Editing
Proportional Editing: Disable
Proportional Editing Falloff: Smooth
Proportional Size: 1.000
Confirm on Release



Resize
Vector: X: 2.000, Y: 2.000, Z: 2.000
Constraint Axis: X, Y, Z
Orientation: Global
Mirror Editing
Proportional Editing: Disable
Proportional Editing Falloff: Smooth
Proportional Size: 1.000
Edit Object data texture
Confirm on Release

Redo Last - Rotation (Object Mode, 60 degrees) __________ Redo Last - Scale (Edit Mode, Resize face)

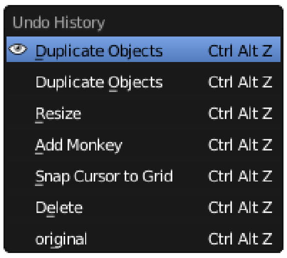
Operations using Redo Last

Some operations produce particularly useful results if you tweak their parameters with the F6 Menu. Take, for example, adding a Circle. If you reduce the Vertex count to 3, you get a perfect equilateral triangle.

Undo History

Mode: All modes

Hotkey: CtrlAltZ



Undo History	
Duplicate Objects	Ctrl Alt Z
Duplicate Objects	Ctrl Alt Z
Resize	Ctrl Alt Z
Add Monkey	Ctrl Alt Z
Snap Cursor to Grid	Ctrl Alt Z
Delete	Ctrl Alt Z
original	Ctrl Alt Z

The Undo History menu, which appears upon CtrlAltZ press.

There is also a Undo History of your actions, recorded by Blender. You can access the history with CtrlAltZ.

Rolling back actions using the *Undo History* feature will take you back to the action you choose. Much like how you can alternate between going backward in time with CtrlZ and then forward with ⇧ ShiftCtrlZ, you can hop around on the Undo timeline as much as you want as long as you do not make a new change. Once you do make a new change, the Undo History is truncated at that point.

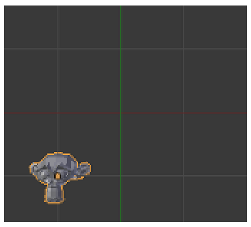
Repeat Last

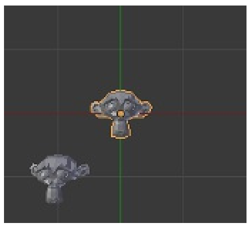
Mode: All modes

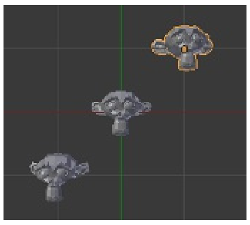
Hotkey: ⇧ ShiftR

The Repeat Last feature will Repeat your last action when you press ⇧ ShiftR.

In the example Images below, we duplicated a *Monkey* Mesh, and then we moved the Object a bit. Using repeat ⇧ ShiftR, the *Monkey* was also duplicated and moved.


Suzanne.

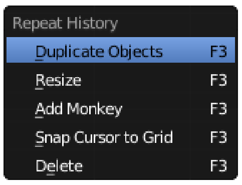

After a ⇧ ShiftD and move.


After a ⇧ ShiftR.

Repeat History

Mode: All modes

Hotkey: F3

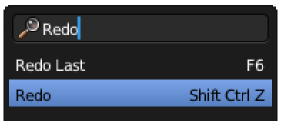


The Repeat menu, which appears upon F3 press.

The (New feature) Repeat History will present you a list of the last repeated actions, and you can choose the actions you want to repeat. It works in the same way as the Undo History, explained above, but the list contains only repeated actions. To access Repeat History, use F3.

There are two separate Histories for Blender
Blender uses two separate Histories, one dedicated for the Edit Mode, and one dedicated for the Object Mode.

Blender Search



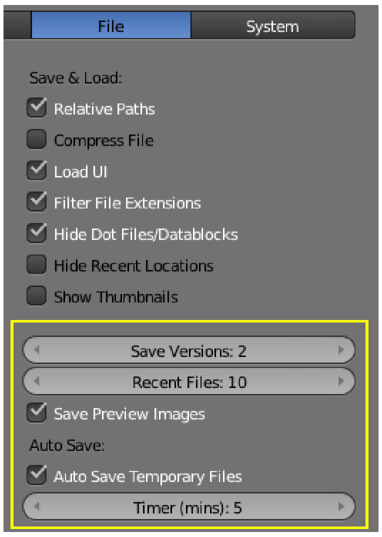
Spacebar search for Redo Last

You can always access all of the explained options for user actions, using Blender Search Space.

Important Note
When you quit Blender, the complete list of user actions will be lost, even if you save your file before quitting.

Options for Files (System Level)

Save and Auto Save



File | System

Save & Load:

- ✓ Relative Paths
- Compress File
- ✓ Load UI
- ✓ Filter File Extensions
- ✓ Hide Dot Files/Datablocks
- Hide Recent Locations
- Show Thumbnails

Save Versions: 2

Recent Files: 10

- ✓ Save Preview Images

Auto Save:

- ✓ Auto Save Temporary Files

Timer (mins): 5

Auto Save options

Computer crashes, power outages or simply forgetting to save can result in the loss or corruption of your work. To reduce the chance of losing files when those events occur, Blender can use an Autosave function. The File tab of the User Preferences window allows you to configure the two ways that Blender provides for you to regress to a previous version of your work.

Save on Quit
: The function Save on Quit is enabled by default in Blender. Blender will always save your files when you quit the application under normal operation.

Save Versions
: This option tells Blender to keep the indicated number of saved versions of your file in your current working directory when you manually save a file. These files will have the extension: `.blend1`, `.blend2`, etc., with the number increasing to the number of versions you specify. Older files will be named with a higher number. e.g. With the default setting of **2**, you will have three versions of your file: `*.blend` (your last save), `*.blend1` (your second last save) and `*.blend2` (your third last save).

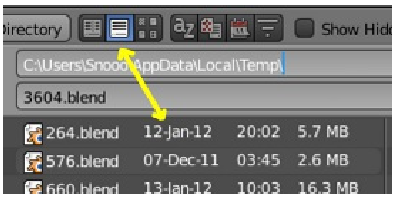
Auto Save Temporary Files
: Checking this box tells Blender to *automatically* save a backup copy of your work-in-progress to the Temp directory (refer to the File panel in the User Preferences window for its location). This will also enable the Timer(mins) control which specifies the number of minutes between each Auto Save. The default value of the Blender installation is **5** (5 minutes). The minimum is **1**, and the Maximum is **60** (Save at every one hour).The Auto Saved files are named using a random number and have a `.blend` extension.

Compress Files

The option to Compress files will try to compact your files whenever Blender is saving them. Large Scenes, dense Meshes, big Textures or lots of elements in your Scene will result in a big `.blend` being created. This option could slow down Blender when you quit, or under normal operation when Blender is saving your backup files. In fact, using this option you will trade processor time for file space.

Recovering Auto Saves

Recover Last Session
: File » Recover Last Session will open the `quit.blend` that is saved into the Temp directory when you exit Blender. Note that files in your Temp directory are deleted when you reboot.



C:\Users\Snood\AppData\Local\Temp\

3604.blend

264.blend	12-Jan-12	20:02	5.7 MB
576.blend	07-Dec-11	03:45	2.6 MB
660.blend	13-Jan-12	10:03	16.3 MB

Blender File Browser

- A Tip: When recovering files, you will navigate to your temporary folder. It is important, when browsing, to enable the detailed list view. Otherwise, you will not be able to figure out the dates of the auto-saved .blends. (See Figure: Blender File Browser)

Recover Auto Save
: File » Recover Auto Save... allows you to open the Auto Saved file. After loading the Auto Saved version, you may save it over the current file in your working directory as a normal `.blend` file.

Important Note

When recovering an Auto Saved file, you will lose any changes made since the last Auto Save was performed.

Only **one** Auto Saved file exists for each project (i.e. Blender does not keep older versions – hence you won’t be able to go back more than a few minutes with this tool).

Other options

Recent Files
: This setting controls how many recent files are listed in the File » Open Recent sub-menu.

Save Preview Images
: Previews of images and materials in the File Browser window are created on demand. To save these previews into your `.blend` file, enable this option (at the cost of increasing the size of your .blend file).

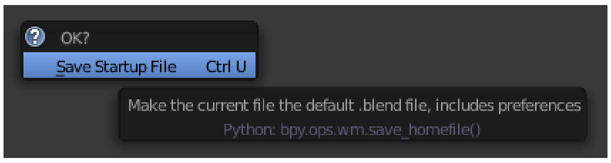
Setting the default Scene

Mode: All modes

Hotkey: CtrlU

Menu: File » Save Startup File

When you start Blender or start a new project with the menu entry File » New or using the shortcut CtrlN, a new Scene is created from the default Scene stored in the Blender install directory and it includes the default User Preferences. This default Scene could instead be another .blend file stored outside of Blender's default install directory. You can save user preferences to the default Scene that comes with Blender, or use another .blend file as a startup file with your customized user preferences.



Make the current file the default .blend file, includes preferences
Python: bpy.ops.wm.save_homefile()

Save User Settings popup

To change the default scene, make all of the desired changes to the current scene or current file and press CtrlU. Note that if you are using another .blend file when you press CtrlU, this file will be the default startup file instead of the one that comes with the default Blender install.

The Save Startup File popup confirmation will appear. Click LMB on the Save Startup File popup or press ↵ Enter.

Press Esc to abort.

Restoring the Default Scene to Factory Settings

Mode: All modes

Hotkey: Undefined, you can add one for your Keymap »

Menu: File » Load factory Settings

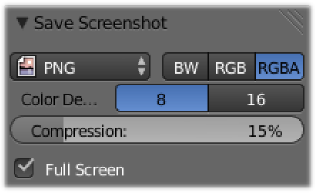
To restore the default scene to the factory settings, LMB in File » Load Factory Settings. This will restore all User Preferences back to the original Factory Settings. To save the changes, use CtrlU and your Factory Settings will be saved as the default Scene for Blender.

User Preferences Window
For more information about the Editor Window for User Preferences or how to clean your preferences manually, please read the chapter about User Preferences

Blender Screenshots

Mode: All modes

Hotkey: CtrlF3

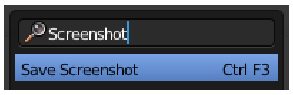


Save Screenshot Option

CtrlF3 will take a screenshot of your Blender window and then open the Blender File Browser window, allowing you to specify the name and location of the screenshot. In the example image at the right, the PNG format will be the output of the screenshot taken (settings are the same as the ones available to save render results). When the Blender File Browser window opens for you, at the left, there is a tab called *Save Screenshot* where you can find format settings and a checkbox with the option *Full Screen*.

- Check the Option to save the entire Blender window (full width and height of the Blender window you are using when you call the command).
- Uncheck the box to save only your active window (where your mouse is located when you call the command).

Keyboard Shortcut Conflicts



Search Functionality

Sometimes, the operating System you are using is designed to use some Shortcuts that the default Blender installation also uses for its functions. In this case, you can use the search functionality present in Blender. (See Fig: Search Functionality). Hit Space and type Screenshot, in the Search Popup

Operating System Screenshots

You may also use the Operating system to capture the screen to the clipboard. You can then paste the image from the clipboard into your image editor.

Windows Screenshots

Press AltPrint screen to capture the active program window to the clipboard.

Mac OSX Screenshots

Press ⌘ Cmd⇧ Shift3 to capture the screen to a file on the desktop.

Press Ctrl⌘ Cmd⇧ Shift3 to capture the screen to the clipboard.

Press ⌘ Cmd⇧ Shift4 to capture an area of the screen to a file on the desktop.

Press Ctrl⌘ Cmd⇧ Shift4 to capture an area of the screen to the clipboard.

GNU/Linux Screenshots

On some Linux distributions (such as Ubuntu) and window managers, you can press Print screen to capture the screen to a file. For other distributions or window managers you may require additional software. Examples of such software include, but are not limited to: xvidcap, scrot and recordMyDesktop. Consult your distribution's manual or software repository for more information.

Software Screenshots

In addition to the options present in Blender and in your Operating System, there is other useful software to take Screenshots of your screen, like Gimp, Photoshop, Screenhunter, and so on.

Gimp Screenshots

Taking Screenshots from Gimp:

- Go to File -> Create -> Screenshot.
- There are two options:
    - Take a Screenshot of a single Window
    - Take a Screenshot of the entire Screen

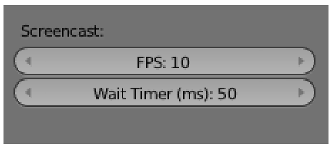
There is also a Delay field, where you can input some delay in seconds. Choose the appropriate options and click on the *Snap* Button. If you choose to Take a Screenshot of a single Window, you will have to click in a Window at the end of the delay.

Blender Screencasts

Mode: All modes

Hotkey: AltF3

The shortcut AltF3 starts the screencast function. Screencasts will record your actions over time either as a video or sequence of image files. The type and location of the output is determined by the settings in the Output panel of the Render context window. The default settings will generate a screencast consisting of a series of PNG images captured every 50 ms and stored in the /tmp folder. If you want to record a video, set the Output to one of the Movie File Formats supported by your system listed in the Output panel format menu. If you are unsure what video codecs your system supports, select AVI JPEG.


Screencast:
FPS: 10
Wait Timer (ms): 50

Options in the User Preferences Editor

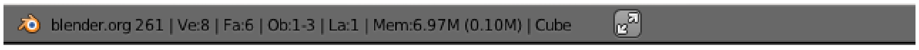
The FPS for video Screencasts and time between each Screenshot for an image series Screencast can be set from the System panel of the User Preferences window.

(See Fig: Options in the User Preferences Editor)

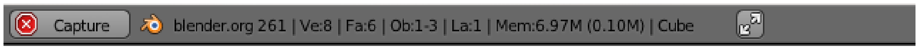
Audio support
Blender Screencast doesn't support audio recordings, so you will have to do it manually using other software, e.g. Audacity, in conjunction with Blender.

When you start Blender Screencasts, the header of the Info Window will change, and it will show you a button for stopping your capture.

Below, we show the normal header of the Info Window, when in normal Blender operation (See Fig: Info Window - Header - Normal Operation), and with the Stop button for the Screencast, when in Screencast Mode. (See Fig: Info Window - Header - Capture Stop Button).


blender.org 261 | Ve:8 | Fa:6 | Ob:1-3 | La:1 | Mem:6.97M (0.10M) | Cube

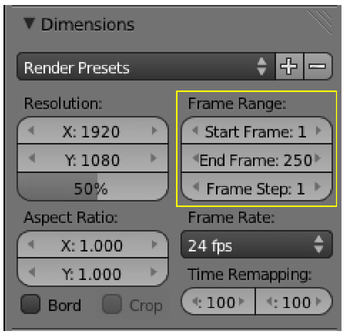
Info Window - Header - Normal Operation


Capture blender.org 261 | Ve:8 | Fa:6 | Ob:1-3 | La:1 | Mem:6.97M (0.10M) | Cube

Info Window - Header - Capture Stop Button

(Note: The header Image was taken using Blender 2.61)

The only way to stop the Screencast
Pressing the Stop button in the header of the Info Window is the only way to stop the Screencast capture. If you press Esc, the shortcut will only work for operations performed in the Blender User Interface, (it will stop animations, playbacks and so on...), but will not work to stop Screencasts.


▼ Dimensions
Render Presets
Resolution:
X: 1920
Y: 1080
50%
Aspect Ratio:
X: 1.000
Y: 1.000
Bord
Crop
Frame Range:
Start Frame: 1
End Frame: 250
Frame Step: 1
Frame Rate:
24 fps
Time Remapping:
:100
:100

Dimensions Panel - Frame Range

The frames are stored using a suffix added to their file name, where the suffix is composed of the numbers present in the fields for *start* and *end frames*, defined in the Frame Range of the Dimensions panel, Render context. (See Fig: Dimensions Panel - Frame Range - highlighted in yellow)

Important:

The configuration of the End frame, present in the Frame Range of the Dimensions Panel, **will not** stop your capture automatically. You will always have to stop the Screencast manually, using the Stop button.

The Videos are generated internally in the same manner as the Screenshots, using the width and height of the Window you are working in. If you choose to capture to a Video file, Blender will have to pass those frames to a Video codec.

Warning: Some codecs limit the output width/height or the video quality.

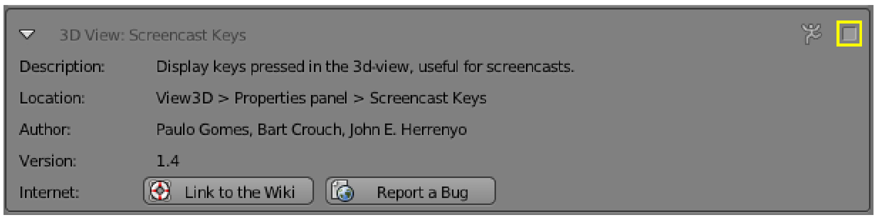
- When you save your Screencast in an Image format, the Images will be saved using the entire Blender Window, with full width and height, and the quality of the Image will be defined by its type (i.e. JPG, PNG, and so on) and configuration (i.e. Slider *quality* of the .JPG format).
- When you save your Screencast in a Video format, it will be sent to a codec. Depending on the codec limitations, the resulting output Video could be scaled down. Furthermore, some combinations of Window width and height cannot be processed by certain codecs. In these cases, the Screencast will try to start, but will immediately stop. In order to solve this, choose another Window format and/or another codec.

Blender Window Dimension

There is a way to match the Blender Window dimensions with the Output Video File, achieving standard dimensions for the output of the Blender Screencast. (I.e. NTSC, HD, Full HD, etc). You can control the width and height of your Blender Window, starting Blender from a Command Line. To learn more about starting Blender from a command line, see the page about Blender Console Window.

Addon: 3D View:Screencast Keys

The community based Addon 3D View:Screencast Keys will show you the keys, combination of keys pressed and mouse clicks on the left bottom corner of your 3D screen every time you press a key or mouse button when capturing Screencasts. The community Addon comes with the default installation of Blender. The Image below shows the community Addon with its Tab Open. (See Fig: 3D View: Screencast Keys - Addon). To enable the Addon, open the User Preferences Editor Window CtrlAltU, go to the Addons Tab, and go to the *3D View*Addons. Just click on the checkbox (Highlighted in yellow) to enable the Addon.



3D View: Screencast Keys

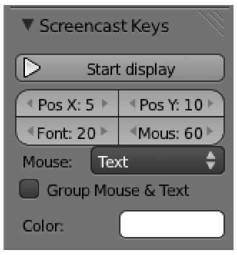
Description: Display keys pressed in the 3d-view, useful for screencasts.
Location: View3D > Properties panel > Screencast Keys
Author: Paulo Gomes, Bart Crouch, John E. Herrenyo
Version: 1.4
Internet: Link to the Wiki | Report a Bug

3D View: Screencast Keys - Addon

Mode: All modes → Addon Enabled

Hotkey: Use N to show the Properties Panel → Screencast Keys Tab

Menu: View » Properties → Screencast Keys Tab



Screencast Keys Addon Tab - Properties Panel

Once the Addon is enabled you will see the Screencast Keys section at the end of the list, on the Properties panel.

Description:

- **Start display button:** When you press this button, Blender will display any Key or combination of Keys you are pressing on the bottom left corner of the 3D window as floating text. If you press several times the same Key or combination of Keys, Blender will add an " xn" tag at the end of the Keys or combination of Keys, indicating how many times you pressed the Key or combination of Keys.
- **Stop display button:** will stop Blender from displaying ScreenCast Keys.
- **PosX:** postion of the Screencast text on **X** axis.
- **PosY:** position if the Screencast text on **Y** axis.
- **Font:** Screencast text font size.
- **Mouse:** Screencast mouse icon size.
- **Mouse display:** In this drop down menu you can select how the Screencast text will be displayed
- **Text:** Will display the Keys pressed and Mouse buttons pressed as text.
- **Icon:** Will display the Mouse as an icon and Keys pressed as text.
- **None:** Will display info about Keys pressed only, without mouse button info.

- **Group Mouse & Text Check box:** When this is checked, Blender will display a box around the Screencast Text to make reading easy.
- **Color:** Lets you choose the color of the Screencast text.

New Community Addon

There is also currently an Addon for Blender 2.5/2.6 which will take a screenshot of any area you like at the click of a button, and proceed to upload it directly to Pasteall. The Addon currently has no development page, but it will be linked to here when it's finished.

Help system

Mode: All modes

Hotkey: Undefined - You can add one for your Keymap »

Menu: Help

Blender has a range of built-in and web-based Help options.

The built in help options include:

- A Menu with all of the Help Options including the Web based ones. Some of them are also present in the Splash Screen.

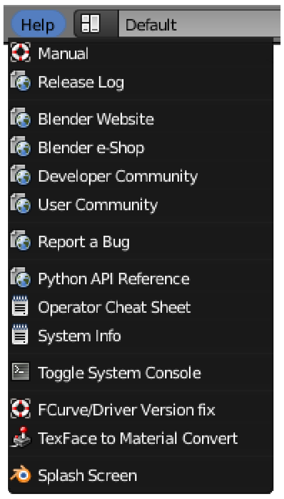
Other new features like:

- The Blender Search (new feature).
- Tooltips showing also the internal Python Operators (new feature), when the user hovers the Mouse over a Button, a Menu, Numeric Field or any Blender function that has a named Python Operator.

General Web-based Help Options

Browser and Internet Connection

Some forms of Help start up your web browser and access the Blender Foundation's web servers. In order to do this, you must have configured a default web browser for your Operating System, and have a connection to the Internet.



Help menu

- *Manual* - This is a link for the Official Blender Manual, in *Wiki* format, which you are now reading.
- *Release Log* - The release notes on the Web for the current Blender version.
- *Blender Website* - The blender.org home page.
- *Blender e-Shop* - The Blender e-Store, where you can buy Training DVD's, books, t-shirts and other products.
- *Developer Community* - The blender.org "Get Involved" page. This is the launch page for Blender software development, bug tracking, patches and scripts, education and training, documentation development and functionality research.
- *User Community* - Lists of many different support venues here.
- *Report a Bug* - The Blender Bug Tracker page.

Important: in order to Report a Bug, you must register at the website.

Programming Options

- *Python 2.6X API Reference* - Python application programming interface (API) that Blender and Python use to communicate with each other. Useful for the Blender Game Engine, Customizing, and other scripting.

Operator Cheat Sheet

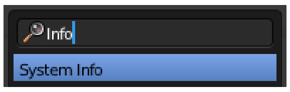
- Operator Cheat Sheet - Creates the `OperatorList.txt` file, which you can access in the Text Editor. You can also use Blender Search to generate the file. The text will list the available Python operators. At the time we were writing this part of the Manual (Blender 2.61), Blender had 1245 Operators.

While Blender is generating this list, the Info Window will change, showing a message for the operation (See Fig: Info Window – Operator Cheat Sheet). To read the Text, switch to the Blender Text Editor Window, using the Window type Selector, and then, clicking on the button *Browse Text to be Linked* of the Text Editor, your text block will be shown in the Editor. The file will be in your list of Text files, named as *OperatorsList.txt*, if the file is already generated, Blender will add a numeric suffix for the subsequent ones.

See OperatorList.txt textblock
blender.org 261 | Ve:0 | Fa:0 | Ob:0-2 | La:1 | Mem:7.24M (0.10M)

Info Window – Operator Cheat Sheet

Diagnostics Options



Blender Search - System Info

- System Info - Creates a `system-info` file, which you can access in the Blender Text Editor. The text lists various key properties of your system and Blender, which can be useful in diagnosing problems. When you click on this Option, Blender will verify your installation, will change the Info Window for a while when generating the file (See: Info Window – Info.txt). You can also use Blender Search to generate the file.

To read the Text, switch to the Blender Text Editor Window, using the Window type Selector, and then, clicking on the button *Browse Text to be Linked* of the Text Editor, your text block will be shown in the Editor. The file will be in your list of Text files, named as *system-info.txt*, if the file is already generated, Blender will add a numeric suffix for the subsequent ones.

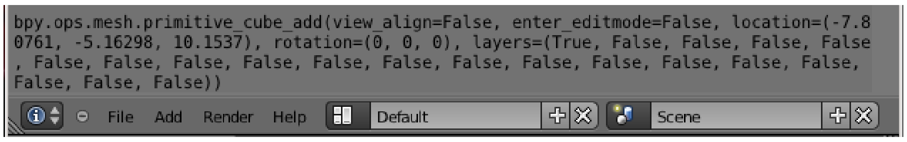
- The text file is created with **4** different sections: Blender, Python, Directories and OpenGL, which we will explain below:
    - **Blender:** This section of the info.txt shows you the Blender version, flags used when Blender was compiled, day and time when Blender was compiled, build system, and the path in which Blender is running.
    - **Python:** The Python version you are using, showing the paths of the Python programming language paths.
    - **Directories:** The Blender directories setup for `scripts`, `user scripts`, `datafiles`, `config`, `scripts (internal)`, `autosave` directory and `temp dir`. Those directories are configured using the User Preferences Editor Window.
    - **OpenGL:** This section will show you the version of OpenGL that you are using for Blender, the name of the manufacturer, version, vendor and a list with your card capabilities or OpenGL software capabilities.

System information generated in 'system-info.txt'
blender.org 261 | Ve:0 | Fa:0 | Ob:0-2 | La:1 | Mem:6.89M (0.1

Info Window – Info.txt

- Toggle System Console - Reveals the command window that contains Blender's stdout messages. Can be very useful for figuring out how the UI works, or what is going wrong if you encounter a problem. Even more information is available here, if you invoke Blender as blender -d. This menu item only shows up on Windows.
    - In all Operating Systems, to see this information, simply run blender from the command-line.
    - On Linux, if you ran Blender from the GUI, you can see the output in ~/.xsession-errors
    - On Mac OS X, you can open Console.app (in the Utilities folder in Applications) and check the Log there.

- Info Window Log - This is not exactly a Help menu, but it is related. If you mouseover the line between the Info window and the 3D then click and drag the Info window down a bit, you can see the stream of Python calls that the UI is making when you work. This can be useful in creating scripts.


```
bpy.ops.mesh.primitive_cube_add(view_align=False, enter_editmode=False, location=(-7.8
0761, -5.16298, 10.1537), rotation=(0, 0, 0), layers=(True, False, False, False, False
, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False))
```

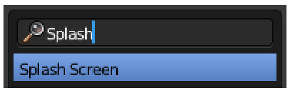
The Info Window Log after adding a Cube

Legacy Version Support

- FCurve/Driver fix - Sometimes, when you load .blend's made from older versions of Blender (2.56 and previous), the Function Curves and Shapekey Drivers will not function correctly due to updates in the animation system. Selecting this option updates the FCurve/Driver data paths.

- TexFace to Material Convert - Convert old Texface settings into material. It may create new materials if needed.

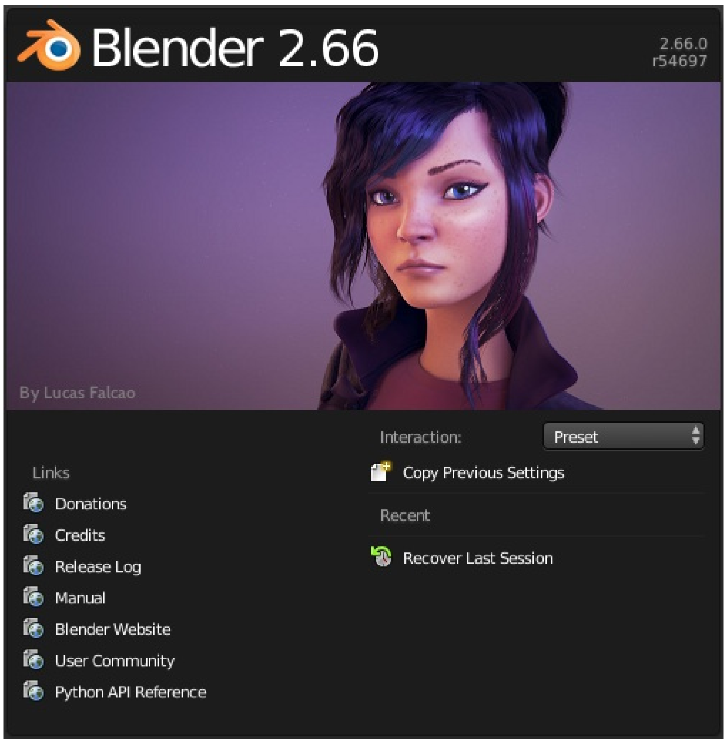
Splash Screen



Splash Screen Search

Splash Screen - This displays the image where you can identify package and version. At the top-right corner, you can see the Version and SVN (Subversion) revision (See Fig: Blender Splash Screen). For example, in our Splash Screen, you can see the version **2.66.0** and the revision number **r54697**. This can be useful to give to support personnel when diagnosing a problem. You can also use Blender Search to Show the Splash Screen or click in the Small Blender Logo present in the Info Window

There are some Internet Based Help options that are also present in the Blender Splash Screen. They are presented as the same links you will find at the Help Menu.



Blender Splash Screen, Blender Version 2.66

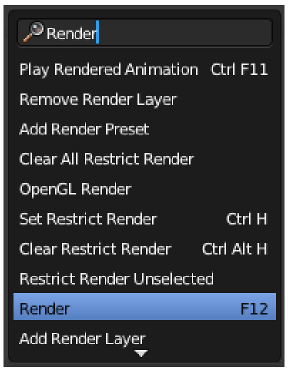
Other Help Options

Here we explain the two new features added for Blender, Blender Search and the recoded Tooltips.

Blender Search

Mode: All modes

Hotkey: Space

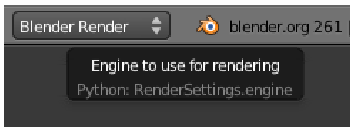

Render
Play Rendered Animation Ctrl F11
Remove Render Layer
Add Render Preset
Clear All Restrict Render
OpenGL Render
Set Restrict Render Ctrl H
Clear Restrict Render Ctrl Alt H
Restrict Render Unselected
Render F12
Add Render Layer
Blender Search - Render

The Blender Search feature, called Blender Search, is a new functionality added by the Blender recode (from 2.4x series to 2.5x series and so on). The Internal name of the feature is *Operator Search*. When you hit Space from your keyboard, Blender will present you with a small Pop Up Window, no matter which Blender Window your Mouse pointer is located (except the Text Editor Window and Python console), and a field for you to type in. Just type what you need and Blender will present you a list of available options. You can click on the appropriate function for you, or search through them using your keyboard, type ↵ Enter to accept, or Esc to leave. Clicking outside of the Blender Search Window or taking the Mouse pointer away, will also leave Blender Search.

The Image at the right shows Blender Search when we type the word *Render* inside the field. If you continue typing, your search keywords will refine your search and if no named operator can be found, the small Pop Up Window for the Blender Search will stay blank.

- How it works:
    - Every Blender Internal Operator can use a defined name, some of them are predefined names for the user. For example, the Render command is a named Python call, the appropriate Operator is `Python: bpy.ops.render.render()` , but for the user, it is called Render. All of those *user* names that were previously attributed for Python operators can searched for using Blender Search.

Tooltips



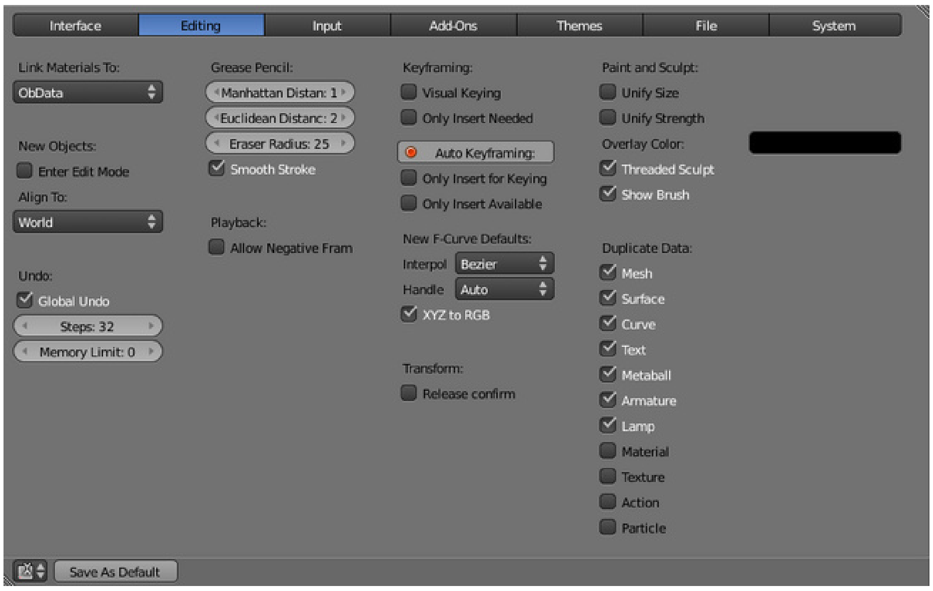
Engine to use for rendering
Python: RenderSettings.engine

The Mouse pointer was Stopped for a while over the Render Engines List in the Info Window. The normal Tooltip is in white and the Python operator is displayed in grey

The Tooltips in Blender were completely recoded, and every time you hover your Mouse over a Button, a Command, Numeric Fields or things that are related to Operators, staying for a while, it will show you not only the normal Tooltip, but also the specific related operator. Those operators are useful for lots of tasks, from Python Scripts to Keymaps. In the example Image at the right, we pointed our Mouse over the Info Window, specifically over the list of the Render engines available, waited for a while, and the Tooltip with the appropriate operator was shown. In our example, it shows the Tooltip *Engine to Use for Rendering* in white, and `Python: RenderSettings.engine` in grey, which is the Operator associated with the function.

Open User Preferences

To open a Blender User Preferences editor go to File » User Preferences or press CtrlAltU. Mac users can press ⌘ Cmd,. You can also load the Preferences editor in any window by selecting User Preferences from the Window type selection menu.


Interface | Editing | Input | Add-Ons | Themes | File | System

Link Materials To: ObData

New Objects:
☐ Enter Edit Mode

Align To: World

Undo:
☑ Global Undo
Steps: 32
Memory Limit: 0

Grease Pencil:
Manhattan Distan: 1
Euclidean Distanc: 2
Eraser Radius: 25
☑ Smooth Stroke

Playback:
☐ Allow Negative Fram

Keyframing:
☐ Visual Keying
☐ Only Insert Needed
● Auto Keyframing:
☐ Only Insert for Keying
☐ Only Insert Available

New F-Curve Defaults:
Interpol: Bezier
Handle: Auto
☑ XYZ to RGB

Transform:
☐ Release confirm

Paint and Sculpt:
☐ Unify Size
☐ Unify Strength

Overlay Color: ■
☑ Threaded Sculpt
☑ Show Brush

Duplicate Data:
☑ Mesh
☑ Surface
☑ Curve
☑ Text
☑ Metaball
☑ Armature
☑ Lamp
☐ Material
☐ Texture
☐ Action
☐ Particle

Save As Default

This editor permits you to configure how Blender will work. The available options are grouped into seven tabs, accessible at the top of the window. The options are: *Interface*, *Editing*, *Input*, *Add-Ons*, *Themes*, *File* and *System*.

Configure

Now that you have opened the User Preferences editor, you can configure Blender to your liking. Select what you want to change in the following list:

Interface • Editing • Input • Add-Ons • Themes • File • System

Save the new preferences

Once you have set your preferences, you will need to manually save them, otherwise the new configuration will be lost after a restart. Blender saves its preferences to userpref.blend in your user folder.

In the User Preferences window, click on Save User Settings. This will save all of the new preferences.

Load Factory Settings

There are two ways to restore the default Blender settings:

1. Go to File » Load Factory Settings and then save the preferences with CtrlU or via the User Preferences editor.
2. Delete the `startup.blend` file from the following location on your computer:
   - Linux: */home/$user/.blender/'Version Number'/config/startup.blend* (you'll need to show hidden files).
   - Windows 7 and Windows Vista: *C:\Users\$user\AppData\Roaming\Blender Foundation\Blender\'Version Number'\config\startup.blend'*
   - MacOS: */Users/$user/Library/Application Support/Blender/'Version Number'/config/startup.blend* (you'll need to show hidden files).

```
While you're in the Blender config folder, it can be valuable to copy your Blender settings file to another
folder. In the event that you lose your configuration, you can restore your Blender settings file with your backup
copy.
```

Display

Tooltips
: When enabled, a tooltip will appear when your mouse pointer is over a control. This tip explains the function of what's under the pointer and the associated hotkey (if any). When disabled, you can hold Alt/Option before moving the mouse pointer over a control to display the tooltip.

Show Python Tooltips
: When enable the python function related to the item will be shown in the tooltip.

Object Info
: Display the active Object name and frame number at the bottom left of the 3D view.

Large Cursors
: Use large mouse cursors when available.

View Name
: Display the name and type of the current view in the top left corner of the 3D window. For example: User Persp or Top Ortho.

Playback FPS
: Show the frames per second screen refresh rate while an animation is played back. It appears in the viewport corner, displaying red if the frame rate set can't be reached.

Global Scene
: Forces the current scene to be displayed in all screens (a project can consist of more than one scene).

Object Origin Size
: Diameter of 3D Object centers in the view port (value in pixels from 4 to 10).

Display Mini Axis
: Show the mini axis at the bottom left of the viewport.

Size
: Size of the mini axis.

Brightness
: Adjust brightness of the mini axis.

View manipulation

Cursor Depth
: Use the depth under the mouse when placing the cursor.

Auto Depth
: Use the depth under the mouse to improve view pan/rotate/zoom functionality.

Zoom to Mouse Position
: When enabled, the mouse pointer position becomes the focus point of zooming instead of the 2D window center. Helpful to avoid panning if you are frequently zooming in and out.

Rotate Around Selection
: The selected object becomes the rotation center of the viewport.

Global Pivot
: Lock the same rotation/scaling pivot in all 3D views.

Auto Perspective
: Automatically to perspective Top/Side/Front view after using User Orthographic. When disabled, Top/Side/Front views will retain Orthographic or Perspective view (whichever was active at the time of switching to that view).

Smooth View
: Length of time the animation takes when changing the view with the numpad (Top/Side/Front/Camera...). Reduce to zero to remove the animation.

Rotation Angle
: Rotation step size in degrees, when 4 NumPad, 6 NumPad, 8 NumPad, or 2 NumPad are used to rotate the 3D view.

2D Viewports

Minimum Grid Spacing
: The minimum number of pixels between grid lines in a 2D (i.e. top orthographic) viewport.

TimeCode Style
: Format of Time Codes displayed when not displaying timing in terms of frames. The format uses '+' as separator for sub-second frame numbers, with left and right truncation of the timecode as necessary.

Manipulator

Permits configuration of the 3D transform manipulator which is used to drag, rotate and resize objects (Size, Handle size).

Menus

Open on Mouse Over
: Select this to have the menu open by placing the mouse pointer over the entry instead of clicking on it.

Menu Open Delay

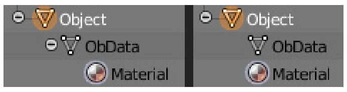
Top Level
: Time delay in 1/10 second before a menu opens (Open on Mouse Over needs to be enabled).

Sub Level
: Same as above for sub menus (for example: File » Open Recent).

Link Materials To

To understand this option properly, you need to understand how Blender works with Objects. Almost everything in Blender is organized in a hierarchy of Datablocks. A Datablock can be thought of as containers for certain pieces of information. For example, the Object Datablock contains information about the Object's location while the Object Data (ObData) datablock contains information about the mesh.

A material may be linked in two different ways:



A material linked to ObData (left) and Object (right).

ObData
: Any created material will be created as part of the ObData datablock.

Object
: Any created material will be created as part of the Object datablock.

Read more about Blender's Data System »

New objects

Enter Edit Mode
: If selected, Edit Mode is automatically activated when you create a new object.

Align To
: World

    New objects align with world coordinates.

  View

    New object align with view coordinates.

Undo

Global Undo
: Works by keeping a full copy of the file in memory (thus needing more memory).

Step
: Number of Undo steps available.

Memory Limit
: Maximum memory usage in Mb (0 is unlimited).

Read more about Undo and Redo options »

Grease Pencil

Grease Pencil permits you to draw in the 3D viewport with a pencil-like tool.

Manhattan Distance
: The minimum number of pixels the mouse has to move horizontally or vertically before the movement is recorded.

Euclidian Distance
: The minimum distance that mouse has to travel before movement is recorded.

Eraser Radius
: The size of the eraser used with the grease pencil.

Smooth Stroke
: Smooths the pencil stroke after it's finished.

Playback

Allow Negative Frame
: If set, negative framenumbers might be used.

Keyframing

In many situations, animation is controlled by keyframes. The state of a value (e.g. location) is recorded in a keyframe and the animation between two keyframes is interpolated by Blender.

Visual Keying
: Use Visual keying automatically for constrained objects.

Only Insert Needed
: When enabled, new keyframes will be created only when needed.

Auto Keyframing

Automatic keyframe insertion for Objects and Bones. Auto Keyframe is not enabled by default.

Only Insert Available

Automatic keyframe insertion in available curves.

New F-Curve Defaults

Interpolation

This controls how the state between two keyframes is computed. Default interpolation for new keyframes is Bezier which provides smooth acceleration and de-acceleration whereas Linear or Constant is more abrupt.

XYZ to RGB

Color for X, Y or Z animation curves (location, scale or rotation) are the same as the colour for the X, Y and Z axis.

Transform

Release confirm

Dragging LMB on an object will move it. To confirm this (and other) transforms, a LMB is necessary by default. When this option is activated, the release of LMB acts as confirmation of the transform.

Sculpt Overlay Color

This color selector allows the user to define a color to be used in the inner part of the brushes circle when in sculpt mode, and it is placed as an overlay to the brush, representing the focal point of the brush influence. The overlay color is visible only when the overlay visibility is selected (clicking at the *eye* to set its visibility), and the transparency of the overlay is controled by the alpha slider located at the brush selector panel, located at the top of the tool shelf, when in sculpt mode.

Duplicate Data

The 'Duplicate Data' check-boxes define what data is copied with a duplicated Object and what data remains linked. Any boxes that are checked will have their data copied along with the duplication of the Object. Any boxes that are not checked will instead have their data linked from the source Object that was duplicated.

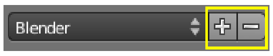
For example, if you have Mesh checked, then a full copy of the mesh data is created with the new Object, and each mesh will behave independently of the duplicate. If you leave the mesh box unchecked then when you change the mesh of one object, the change will be mirrored in the duplicate Object.

The same rules apply to each of the check-boxes in the 'Duplicate Data' list.

Managing presets

Blender lets you define multiple Preset input configurations. Instead of deleting the default keymap to create yours, you can just add new Presets for both the mouse and keyboard. Mouse options can be found on the left hand side of the window and keyboard options to the right in the above picture.

Adding and deleting presets



Before changing anything in the default configuration, click on the "plus" symbol shown in the picture to add a new Preset. Blender will ask you to name your new preset after which you can select the Preset from the list to edit it. If you want to delete your Preset, select it from the list and then click the "minus" symbol.

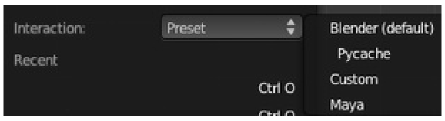
Selecting presets

You can change the preset you are using by doing one of the following:

- Selecting the configuration from the Interaction menu of the splash screen at startup or by selecting Help » Splash Screen.
- Selecting the configuration from the User Preferences Input window.

Note
Note that either of the above options will only change the preset for the current file. If you select File » New or File » Open, the default preset will be re-loaded.

Setting presets to default



Once you've configured your mouse and keyboard Presets, you can make this the default configuration by:

- Opening the User Preferences Input editor and select your presets from the preset list or,
- Selecting your preset configuration from the splash screen.
- Saving your configuration using the Save As Default option from a User Preferences window or by pressing CtrlU.

Export/Import key configuration

In some cases, you may need to save your configuration in an external file (e.g. if you need to install a new system or share your keymap configuration with the community). Simply LMB Export Key Configuration on the Input tab header and a file browser will open so that you can choose where to store the configuration. The Import Key Configuration button installs a keymap configuration that is on your computer but not in Blender.

Mouse

Emulate 3 Button Mouse
It is possible to use Blender without a 3 button mouse (such as a two-button mouse, Apple single-button Mouse, or laptop). This functionality can be emulated with key/mousebutton combos. This option is only available if Select With is set to Right.

Read more about emulating a 3 button mouse »

Continuous Grab
Allows moving the mouse outside of the view (for translation, rotation, scale for example).
Drag Threshold
The number of pixels that a User Interface element has to be moved before it is recognized by Blender.
Select with
You can choose which button is used for selection (the other one is used to place the 3D cursor).
Double Click
The time for a double click (in ms).

Note
If you're using a graphic tablet instead of mouse, and pressure doesn't work properly, try to place the mouse pointer to Blender window and then unplug/replug your graphic tablet. This might help.

Numpad emulation

The Numpad keys are used quite often in Blender and are not the same keys as the regular number keys. If you have a keyboard without a Numpad (e.g. on a laptop), you can tell Blender to treat the standard number keys as Numpad keys. Just check Emulate Numpad.

View manipulation

Orbit Style

Select how Blender works when you rotate the 3D view (by default MMB). Two styles are available. If you come from Maya or Cinema 4D, you will prefer Turntable.

Zoom Style
: Choose your preferred style of zooming in and out with Ctrl MMB

    Scale
    : Scale zooming depends on where you first click in the view. To zoom out, hold Ctrl MMB while dragging from the edge of the screen towards the center. To zoom in, hold Ctrl MMB while dragging from the center of the screen towards the edge.

    Continue
    : The Continue zooming option allows you to control the speed (and not the value) of zooming by moving away from the initial click-point with Ctrl MMB. Moving up from the initial click-point or to the right will zoom out, moving down or to the left will zoom in. The further away you move, the faster the zoom movement will be. The directions can be altered by the Vertical and Horizontal radio buttons and the Invert Zoom Direction option.

    Dolly
    : Dolly zooming works similarly to Continue zooming except that zoom speed is constant.

    Vertical
    : Moving up zooms out and moving down zooms in.

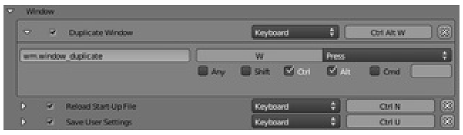
    Horizontal
    : Moving left zooms in and moving right zooms out.

Invert Zoom Direction
: Inverts the Zoom direction for Dolly and Continue zooming.

Invert Wheel Zoom Direction
: Inverts the direction of the mouse wheel zoom.

NDOF device
: Set the sensitivity of a 3D mouse.

Keymap editor


Window
Duplicate Window — Keyboard — Ctrl Alt W
wm.window_duplicate — W — Press
Any — Shift — ✓ Ctrl — ✓ Alt — Cmd
Reload Start-Up File — Keyboard — Ctrl N
Save User Settings — Keyboard — Ctrl U

The Keymap editor lets you change the default Hotkeys. You can change keymaps for each window.

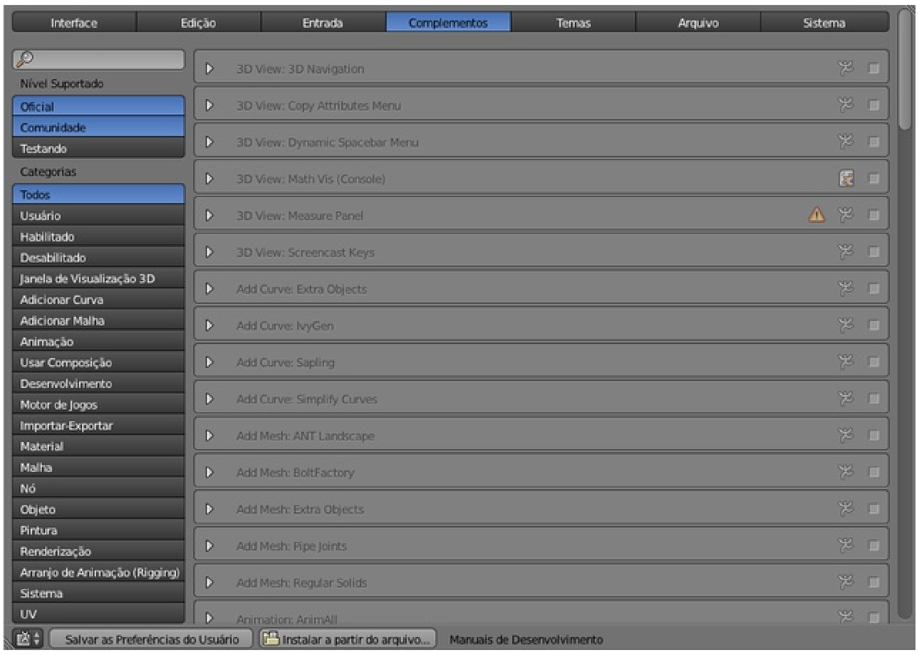
1. Select the keymap you want to change and click on the white arrows to open up the keymap tree.
2. Select which Input will control the function
    - Keyboard: Only hotkey or combo hotkey (E or ⇧ ShiftE).
    - Mouse: Left/middle/right click. Can be combined with Alt, ⇧ Shift, Ctrl, ⌘ Cmd.
    - Tweak: Click and drag. Can also be combined with the 4 previous keys.
    - Text input: Use this function by entering a text
    - Timer: Used to control actions based on a time period. e.g. By default, Animation Step uses Timer 0, Smooth view uses Timer 1.
3. Change hotkeys as you want. Just click on the shortcut input and enter the new shortcut.

If you want to restore the default settings for a keymap, just click on the Restore button at the top right of this keymap.

Interface • Editing • Input • **Add-Ons** • Themes • File • System

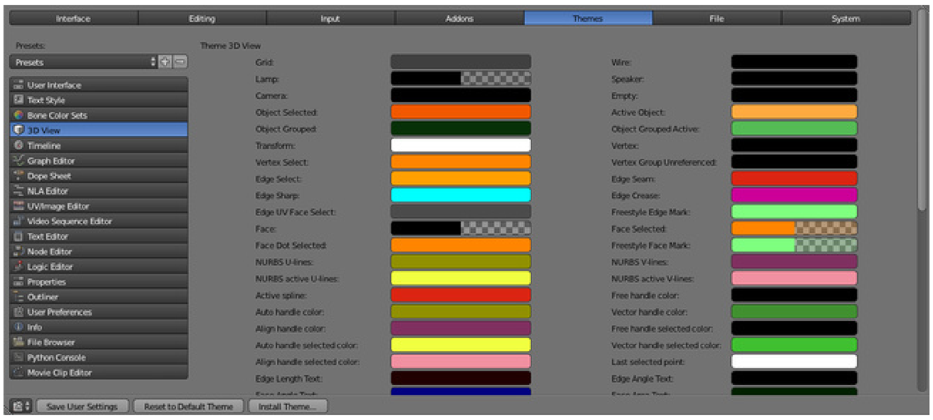
The Add-Ons tab lets you manage secondary options which are not enabled in Blender by default. New features may be added with *Install Add-Ons*. There will be a growing number of such Add-Ons, generated by the Blender-community so look out for that one feature you were missing (or maybe simply create it yourself).

See the Add-Ons Page for more on using Add-Ons.



Customizing themes

As has been said previously, Blender is very customizable—some of the settings affect interface appearance and colors. These are set under the Themes tab.

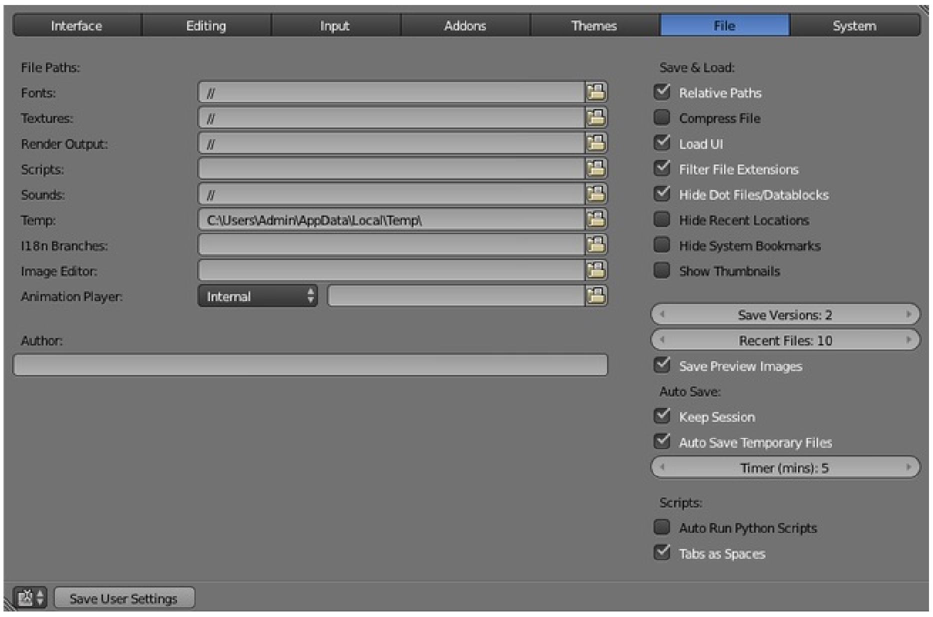


The colors for each editor can be set separately—simply select the editor you wish to change in the multi-choice list at the left, and adjust colors as required. Notice that changes appear in real-time on your screen. In addition, details such as the dot size in the 3D View or the Graph Editor can also be changed.

Themes use blenders preset system, you can save a theme to an XML and install it on another system.

File Preferences

The picture shows the file preferences which are explained below.



File Paths:
Fonts: //
Textures: //
Render Output: //
Scripts:
Sounds: //
Temp: C:\Users\Admin\AppData\Local\Temp\
I18n Branches:
Image Editor:
Animation Player: Internal
Author:

Save & Load:
[✓] Relative Paths
[ ] Compress File
[✓] Load UI
[✓] Filter File Extensions
[✓] Hide Dot Files/Datablocks
[ ] Hide Recent Locations
[ ] Hide System Bookmarks
[ ] Show Thumbnails
Save Versions: 2
Recent Files: 10
[✓] Save Preview Images
Auto Save:
[✓] Keep Session
[✓] Auto Save Temporary Files
Timer (mins): 5
Scripts:
[ ] Auto Run Python Scripts
[✓] Tabs as Spaces

Save User Settings

File Paths

When you work on an important project, it is wise to configure it. Set default paths for the different file types you will be using.

Here is an example of a configuration:

Fonts	//fonts/
Textures	//textures/
Texture Plugins	//plugins/texture/
Sequence Plugins	//plugins/sequence/
Render Output	//renders/
Scripts	//scripts/
Sounds	//sounds/
Temp	//tmp/

Note that blender wont create your project structure automatically. You need to create all directories manually in your file browser.

Scripts Path

By default Blender looks in several directories (OS dependant) for scripts. By setting a user script path in the preferences an additional directory is looked in. This can be used to store certain scripts/templates/presets independently of the currently used Blender Version.

Inside the specified folder specific folders have to be created to tell Blender what to look for where. This folder structure has to mirror the structure of the scripts folder found in the installation directory of Blender:

```
- scripts
   - addons
   - modules
   - presets
       - camera
       - cloth
       - interface_theme
       - operator
       - render
       - ...
   - startup
   - templates
```

Not all of the folders have to be present.

Save & Load

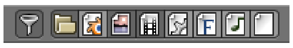
Relative Paths

By default, external files use a relative path. This works only when a Blender file is saved.

Compress File
: Compress `.blend` file when saving.

Load UI

Default setting is to load the Window layout (the Screens) of the saved file. This can be changed individually when loading a file from the Open Blender File panel of the File Browser window.


File extension filter

Filter File Extensions
: By activating this, file dialog windows will only show appropriate files (i.e. `.blend` files when loading a complete Blender setting). The selection of file types may be changed in the file dialog window.

Hide Dot File/Datablocks
: Hide file which start with "**.***" on file browsers (in Linux and Apple systems, "**.***" files are hidden).

Hide Recent Locations
: Hides the Recent panel of the File Browser window which displays recently accessed folders.

Show Thumbnails
: Displays a thumbnail of images and movies when using the File Browser.

Auto Save

Save Versions
: Number of versions created for the same file (for backup).

Recent Files
: Number of files displayed in File » Open Recent.

Save Preview Images
: Previews of images and materials in the File Browser window are created on demand. To save these previews into your `.blend` file, enable this option (at the cost of increasing the size of your `.blend` file).

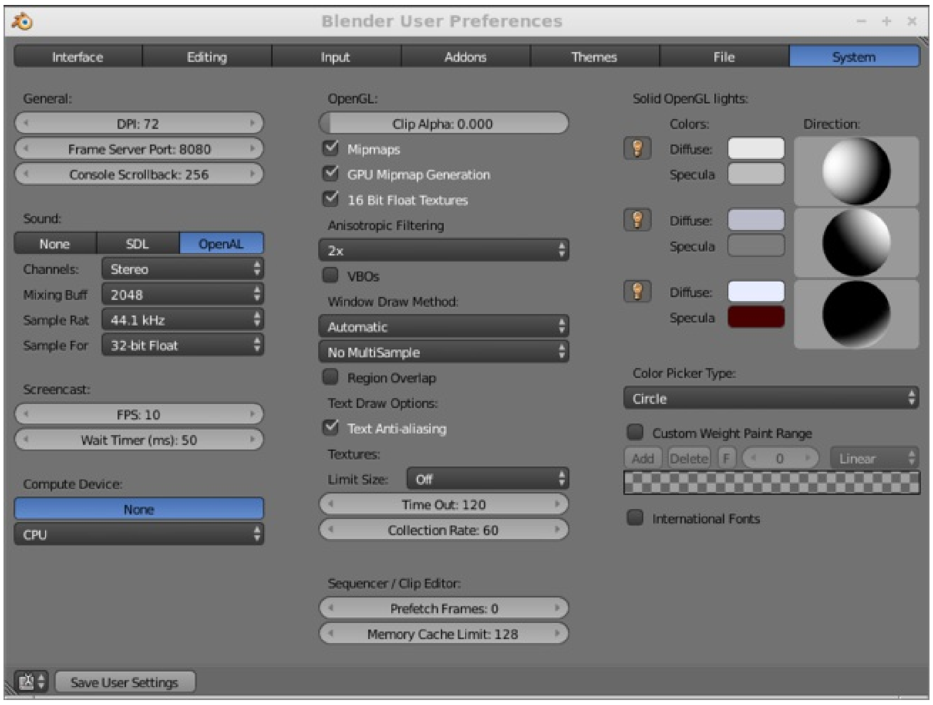
Auto Save Temporary File
: Enable Auto Save (create a temporary file).

Timer
: Time to wait between automatic saves.

Read more about Auto Save options »

System preferences

This picture shows the System tab in the User Preferences editor in Blender. Options are explained below.


Blender User Preferences

Interface | Editing | Input | Addons | Themes | File | System

General:
DPI: 72
Frame Server Port: 8080
Console Scrollback: 256

Sound:
None | SDL | OpenAL
Channels: Stereo
Mixing Buff 2048
Sample Rat 44.1 kHz
Sample For 32-bit Float

Screencast:
FPS: 10
Wait Timer (ms): 50

Compute Device:
None
CPU

OpenGL:
Clip Alpha: 0.000
✓ Mipmaps
✓ GPU Mipmap Generation
✓ 16 Bit Float Textures
Anisotropic Filtering
2x
VBOs
Window Draw Method:
Automatic
No MultiSample
Region Overlap
Text Draw Options:
✓ Text Anti-aliasing
Textures:
Limit Size: Off
Time Out: 120
Collection Rate: 60

Sequencer / Clip Editor:
Prefetch Frames: 0
Memory Cache Limit: 128

Solid OpenGL lights:
Colors: Direction:
Diffuse: Specula
Diffuse: Specula
Diffuse: Specula

Color Picker Type:
Circle
Custom Weight Paint Range
Add Delete F 0 Linear
International Fonts

Save User Settings

General

DPI
: Value of the screen resolution which controls the size of Blender's interface fonts and internal icons shown.
Useful for taking screen shots for book printing and use of high resolution monitors, (such as Retina Display and others) matching the screen DPI with Blender DPI.
During normal usage, most Blender users might prefer to zoom screen elements pressing Ctrl and dragging MMB left and right over a panel to resize its contents,
or use the Numpad+ and Numpad- to zoom in and out the contents.
Pressing ↖ Home (Show All) will reset the zooming at the screen/panel focused by the mouse pointer.
Minimum: **48** , Maximum: **128** , Default:**72**

Frame Server Port
: TCP/IP port used in conjunction with the IP Address of the machine for frameserver rendering. Used when working with distributed rendering.
Avoid changing this port value unless it is conflicting with already existing service ports used by your Operating System and/or softwares.
Always consult your operating system documentation and services or consult your system administrator before changing this value.
Minimum: **0** , Maximum: **32727** , Default: **8080**

Console Scrollback
: The number of lines, buffered in memory of the console window.
Useful for debugging purposes and command line rendering.
Minimum: **32** , Maximum: **32768** , Default:**256**

Sound

Sound
: Set the audio output device or no audio support. There are 3 Options:

None
: No Audio support (no audio output, audio strips can be loaded normally)

SDL
: Uses Simple Direct Media Layer API from libsdl.org to render sounds directly
to the sound device output. Very useful for sequencer strips editing.

OpenAL
: Uses OpenAL soft API for Linux and OpenAL from creative Labs for Windows.
This API provides buffered sound rendering with 3D/spatial support. Useful for the BGE Games.

'Specific sound options' (With SDL or OpenAL enabled)

Channels
Set the audio channel count. Available options are:
Stereo (Default) , 4 Channels , 5.1 Surround , 7.1 Surround
Mixing Buffer
Set the number of samples used by the audio mixing buffer. Available options are:
512 , 1024 , *2048* (Default), 4096 , 8192, 16384, and 32768
Sample Rate
Set the audio sample rate. Available options are:
44.1 Khz (Default), 48 Khs , 96 Khz and 192Khz
Sample Format
Set the audio sample format. Available options are:
32 bit float (Default), 8 bit Unsigned , 16 Bits Signed , 24 Bits Signed , 32 Bits Signed , 32 Bits Float and 64 Bits Float

Screencast

FPS
Framerate for the recorded screencast to be played back.
Minimum: **10** , Maximum: **50** , Default: **10**
Wait Timer
Time in milliseconds between each frame recorded for screencasts.
The default value is **50** milliseconds. Assuming 1 Second = 1000 milliseconds, a fraction of **1/50** is equal to **20** frames per second.
Note that the accuracy of the resulting frames per second for the screencast record will depend on your computer power.
For more Information about Screencasts, please visit *Screencasts* page.
Minimum: **10** , Maximum: **1000** , Default: **50**

Compute Device

The Options here will set the compute device used by the Cycles Render Engine

None
When set to None or the only option is None:
your CPU will be used as a computing device for Cycles Render Engine

When there are other Options for compute device such as
CUDA / OpenCL[1].
If the system has a compatible CUDA enabled graphics card and appropriate device drivers installed.
When one or both of the options are available, the user will be able to choose whether to use CPU or other computing device for Cycles Rendering.

OpenCL[1] *is unsupported, please refer to the* Cycles *Render engine page*

Open GL

Clip Alpha
Clip alpha below this threshold in the 3D viewport.
Minimum: **0.000** (No Clip) , Maximum: **1.000** , Default **0.000** (No Clip)

Mipmaps
Scale textures for 3D view using mipmap filtering. This increases display quality, but uses more memory.

GPU MipMap Generation
Generate MipMaps on the GPU. Offloads the CPU Mimpap generation to the GPU.

16 Bit Float Textures
Enables the use of 16 Bit per component Texture Images (Floating point Images).

Anisotropic Filtering
Set the level of anisotropic filtering. Available Options are:
Off *(No Filtering) , 2x (Default) , 4x , 8x , 16x*

VBOs
Use Vertex Buffer Objects, or vertex arrays if unsupported, for viewport rendering.
Helps to speed up viewport rendering by allowing vertex array data to be stored in Graphics card memory.

Window Draw Method

Window Draw Method
Specifies the Window Draw Method used to display Blender Window(s).

Automatic (Default)
Automatically set based on graphics card and driver.

Triple Buffer
Use a third buffer for minimal redraws at the cost of more memory.
If you have a capable GPU, this is the best and faster method of redraw.

Overlap
Redraw all overlapping regions. Minimal memory usage, but more redraws.
Recommended for some graphics cards and drivers combinations.

Overlap Flip
Redraw all overlapping regions. Minimal memory usage, but more redraws (for graphics drivers that do flipping).
Recommended for some graphic cards and drivers combinations.

Full
Do a full redraw each time. Only use for reference, or when all else fails.
Useful for certain cards with bad to no OpenGL acceleration at all.

Region Overlap
This checkbox will enable Blender to draw regions overlapping the 3D Window.
It means that the Object Tools and Transform Properties Tab,
which are opened by using the shortcuts T and N will be drawn overlapping the 3D View Window.

If you have a capable graphics card and drivers with Triple Buffer support,
clicking the checkbox will enable the overlapping regions to be drawn using the Triple Buffer method,
which will also enable them to be drawn using Alpha, showing the 3D View contents trough the
Object Tools and Transform Properties Tab.

Text Draw Options

Text Draw Options
Enable interface text anti-aliasing.
When disabled, texts are drawn using text straight render (Filling only absolute Pixels).
Default: Enabled.

Textures

Limit Size
Limit the maximum resolution for pictures used in textured display to save memory.
The limit options are specified in a square of pixels, (e.g.: the option 256 means a texture of 256x256 pixels)
This is useful for game engineers, whereas the texture limit matches paging blocks of the textures in the target graphic card memory .
Available Options are:
Off (No limit - Default) , 128 , 256 , 512 , 1024 , 2048 , 4096 , 8192.

Time Out
Time since last access of a GL texture in seconds, after which it is freed. Set to 0 to keep textures allocated.
Minimum: **0** , Maximum: **3600** , Default: **120**

Collection Rate
Number of seconds between each run of the GL texture garbage collector.
Minimum: **0** , Maximum: **3600** , Default: **120**

Sequencer/Clip Editor

Prefetch Frames
Number of frames to render ahead during playback.
Useful when the chosen video codec cannot sustain screen frame rates correctly using direct rendering from the disk to video.
duting video playbacks or editing operations.
Minimum: **0** , Maximum: **500** , Default: **0** (No prefecth)

Memory Cache Limit

Upper limit of the sequencer's memory cache (megabytes).
For optimum clip editor and sequencer performance, high values are recommended.
Minimum: **0** (No cache) , Maximum: **1024** (1 Gigabyte) , Default: **128**

Solid OpenGL lights

Solid OpenGL Lights
: Solid OpenGL Lights are used to light the 3D Window, mostly during Solid view. Lighting is constant and position "world" based.
There are three virtual light sources, also called OpenGL auxiliary lamps, used to illuminate 3D View scenes, which will not display in renders.

The Lamp Icons allows the user to enable or disable OpenGL Lamps.
At least one of the three auxiliary OpenGL Lamps must remain enabled for the 3D View. The lamps are equal, their difference is their positioning and colors.
You can control the direction of the lamps, as well as their diffuse and specular colors. Available Options are:

Direction:
: Clicking with LMB in the sphere and dragging the mouse cursor let's the user change the direction of the lamp by rotating the sphere.
The direction of the lamp will be the same as shown at the sphere surface.

Diffuse:
: This is the constant color of the lamp.
Clicking on the color widget, opens the color picker mini window and allows the user to change colors using the color picker.

Specular:
: This is the highlight color of the lamp
Clicking on the color widget, opens the color picker mini window and allows the user to change colors using the color picker.

Color Picker Type

Color Picker Type
: Choose which type of color dialog you prefer **-** it will show when clicking LMB on any color field.

There are **4** types of color pickers available for Blender

Circle (Default), Square (HS + V) , Square (SV + H) and Square (HV + S)

The color pickers are detailed at the Buttons and Controls page.

Custom Weight Paint Range

Custom Weight Paint Range
: Mesh skin weighting is used to control how much a bone deforms the mesh of a character.
To visualize and paint these weights, Blender uses a color ramp (from blue to green, and from yellow to red).
Enabling the checkbox will enable an alternate map using a ramp starting with an empty range.
Now you can create your custom map using the common color ramp options.
For detailed information about how to use color ramps, please, go to the Buttons and Controls page.

International Fonts

International Fonts
: Blender supports a wide range of languages, enabling this check box will enable Blender to support International Fonts.
International fonts can be loaded for the User Interface and used instead of Blender default bundled font.

This will also enable options for translating the User Interface through a list of languages and Tips for Blender tools which appears whenever the user hovers a mouse over Blender tools.

Blender supports I18N for internationalization, for more information, please, go to the *Internationalization* page.
For more Information on how to load International fonts, please, go to the *Editing Texts* page.

Your First Animation in 30 plus 30 Minutes Part I

This chapter will guide you through the animation of a small "Gingerbread Man" character. We will describe each step completely, but we will assume that you have read the interface chapter, and that you understand the conventions used throughout this book.

In Part I of this tutorial we'll build a *still* Gingerbread Man. Then, in Part II, we will make him walk.

Note
For a much more in-depth introduction to Blender that focuses on character animation, check out the

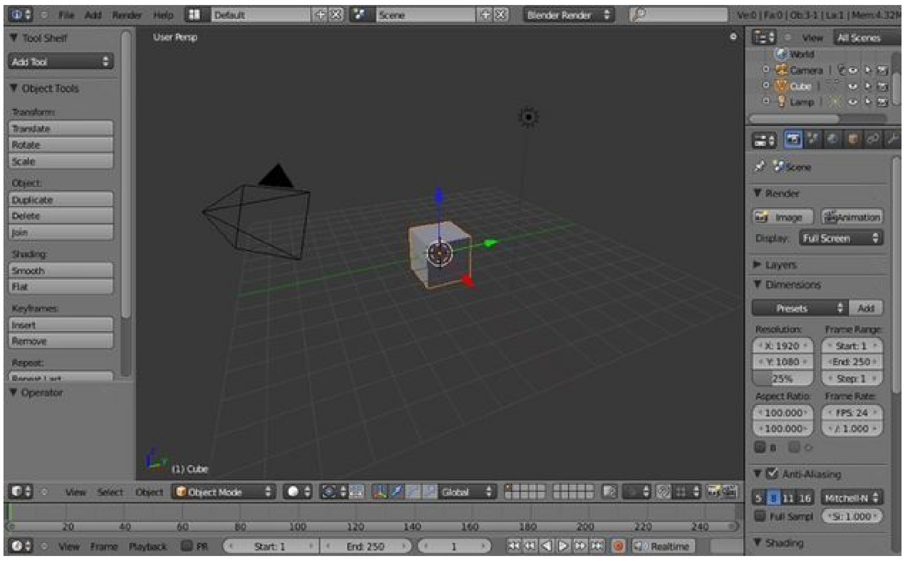
Blender Summer of Documentation Introduction to Character Animation tutorial (this is written for V2.4x but still can be very instructive).

Just like the "Gus the Gingerbread Man" tutorial you see here, the BSoD Intro to Character Animation tutorial assumes no prior knowledge. It guides you through the process of making a walking, talking character from scratch and covers many powerful features of Blender not found here.

The BSoD Intro to Character Animation also has a downloadable PDF version (*3.75 MB*) for offline viewing.

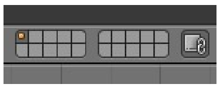
Warming up

After starting Blender, you should see the Default screen set up. The 3D view in the centre displays a camera, a light, and a cube. The cube should already be selected, as indicated by its orange outline.



Default Blender screen.

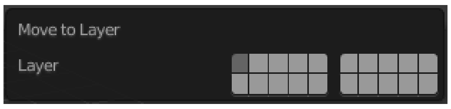
We will start by organizing our working area by placing Objects on different layers. 3D scenes can quickly become confusing when multiple Objects are on the screen. With layers, you can hide Objects you aren't working on and make them visible when you need them.


Layer visibility controls.

Blender provides you with twenty layers to help organize your work. You can see which layers are currently visible from the group of twenty buttons in the 3D window header (see *Layer visibility controls*). You can change the visible layer with LMB and manage the visibility of multiple layers with ⇧ Shift LMB. Visible layers are indicated by a darker gray color in the layer visibility controls. The last layer that is made visible becomes the active layer. The active layer is also where all new Objects will be stored.

Read more about layers »

Let's clean up the screen by first moving the camera and lamp to another layer.



The Move to Layer popup.

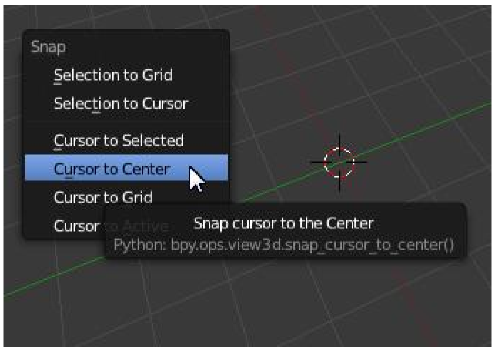
Select the Camera with RMB. Then add the Lamp to the selection with ⇧ Shift RMB. Press M and a small toolbox, like the one in (*The Move to Layer popup*) will appear beneath your mouse with the first button checked. This means that the selected objects are stored in layer 1. Click the rightmost button on the top row. This will move your Camera and Lamp to layer 10.

Keyboard Shortcuts
Blender is mainly controlled with lots of keyboard shortcuts (don't panic, you'll get used to it). Most of those shortcuts only work while the mouse pointer hovers above the corresponding frame. So don't get frustrated if M doesn't do what it's supposed to do -- just move the mouse into the 3D view.

Now make sure that only Layer 1 is visible (colored a darker gray in the layer visibility controls) so that we can start modeling.

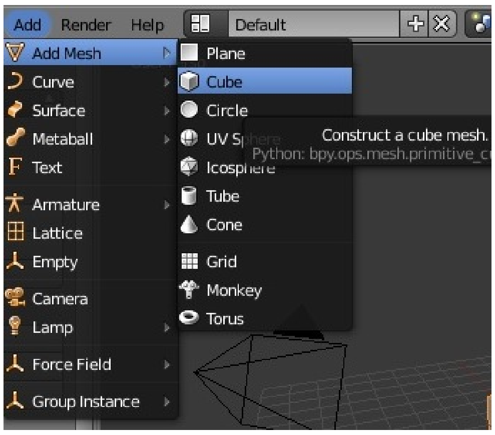
Building the body

With Num lock activated, change to the front view with 1 NumPad and to orthogonal view with 5 NumPad. The upper left corner of the 3D window will tell you whether you are in orthogonal or perspective view.



Snap cursor to the Center

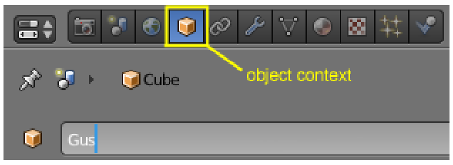
If you don't have a Cube on your screen we'll need to add one. Snap your cursor to the center (0,0,0) of the screen by Object » Snap » Cursor to Center or using the ⇧ ShiftS shortcut.



Add a cube to the scene.

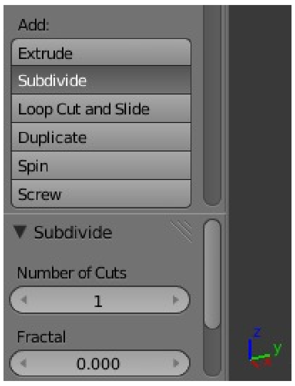
Then add a Cube with Add » Add Mesh » Cube or using the ⇧ ShiftA shortcut. A cube will appear displayed in orange to indicate that it is the active Object. Press ⇆ Tab to enter Edit Mode.

Edit Mode is a mode in which you can edit the vertices of the mesh. By default, all vertices are selected for every new Object created (selected vertices are highlighted in orange, unselected vertices are black). In Object Mode, vertices cannot be selected or edited individually; the Object can be changed only as a whole. You can press ⇆ Tab to switch between these two modes. The current mode is indicated in the header of the 3D window.



Naming Gus

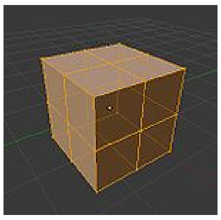
We will call our Gingerbread man "Gus". To do so, switch to the object-context (See *Naming Gus*) which can be found on the Properties Window on the right hand side. You can rename Gus on the first line.



Part of the Tool Shelf

Our first task is to build Gus's body by working on the vertices of our cube. Tools to do this can be found on the Tool Shelf, which is a part of the 3D Window (on the left hand side of the screen). If you can't see the Tool Shelf, simply press T.

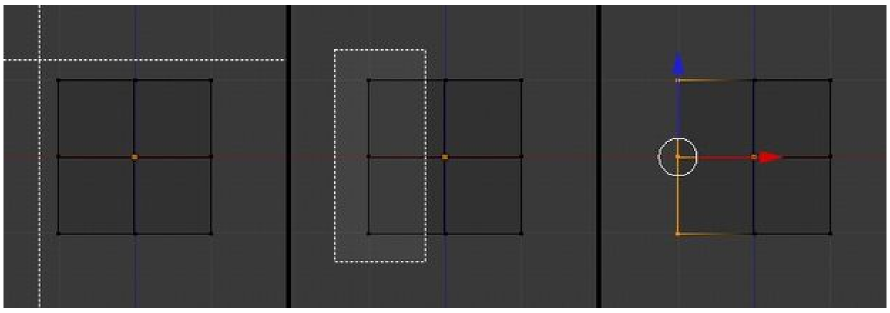
Now locate the Subdivide button in the Tool Shelf (under Add) and press it once. This will split each side of the cube in two, creating new vertices and faces. The result is illustrated below. If you want to get the same view, change to perspective with Num5 and rotate the view by clicking and dragging with MMB. Don't forget to change the view back with Num1 (Front View) and Num5 (Perspective/Orthographic).

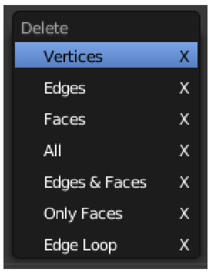


With your cursor hovering in the 3D window, press A to deselect all elements. Vertices will turn black.

You must have the Limit Selection to Visible button *unselected* to continue this tutorial.

Now press B to activate Box Select/Border Select mode. The cursor will change to a couple of orthogonal grey lines. Move the cursor above the top left corner of the cube, press and hold LMB, then drag the mouse down and to the right so that the grey box encompasses all the leftmost vertices. Now release the LMB.


The sequence of Box selecting a group of vertices.



The pop-up menu of the Delete (X) action.

Press X, and from the popup menu select Vertices to erase the selected vertices.

Selection Behavior and Limit Selection to Visible

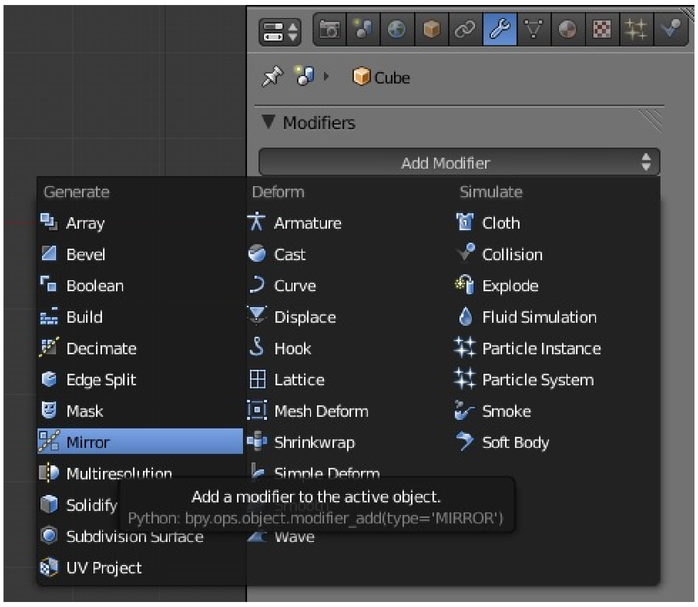
Especially when in orthographic view (toggled via 5 NumPad), there may be vertices hidden behind other vertices. For example, our subdivided cube has 26 vertices, yet in orthographic front view you can only see nine of them because the others are hidden. A RMB click selects only one of these stacked vertices, whereas a box select selects them all. This is true as long as either the Bounding Box or Wireframe display method is active. If the chosen method is Solid or Textured, the Limit Selection to Visible button has to be deactivated or only visible vertices will be selected. (The Limit Selection to Visible button can be found on the 3D window header when either the Solid or Textured display method is active.)

Another tool to select or deselect vertices is Circle Select, which can be activated by pressing C. When the Circle Select tool is active, clicking or dragging with LMB selects vertices. MMB deselects vertices. Using the mouse-wheel changes the size of the selection-circle. RMB or ↵ Enter finalizes the selection and exits Circle Select mode. Just give that alternative a try after you deleted the vertices as mentioned above.

Mirror modelling

To model symmetrical objects we can use the Mirror modifier. It allows us to model only one side of Gus while Blender creates the other in real time. Go to the Properties editor and find the Modifiers context.

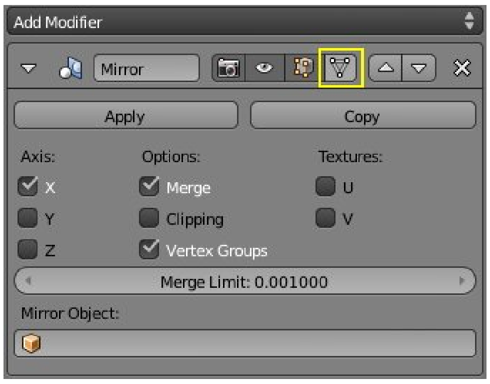
Read more about modifiers »



Add a modifier to the active object.
Python: bpy.ops.object.modifier_add(type='MIRROR')

The modifiers context

It is pretty empty for the moment. Clicking the button marked Add Modifier opens a list where you can choose Mirror.



Cage Mode button.

In addition to affecting Objects in a non-destructive manner, modifiers also allow you to control what is displayed when you are working with them. In our case we will check the Cage Mode button so we can see the transparent mirrored faces in Edit Mode.

We then choose the axis to mirror Gus along by checking either the X, Y or Z button. The mirror plane will be perpendicular to that axis. In our case it is the X-axis.

The Merge button will merge any mirrored vertices that are equal to or closer than the distance specified by the Merge Limit slider. Essentially any mirrored vertex closer to the mirror plane than the limit we set will be placed exactly on the mirror plane and merged with the corresponding vertex. The limit can be set from **0.000** to **1.000** units and how big it should be depends on the nature and the scale of the current job.

For modeling Gus, a vertex that is more than **0.1** units away from the mirror plane would be noticeable but anything closer might not be visible. To prevent a large rip showing up in the middle of our mesh or cause us to neglect a wandering vertex, we should set the Merge Limits to **0.1**.

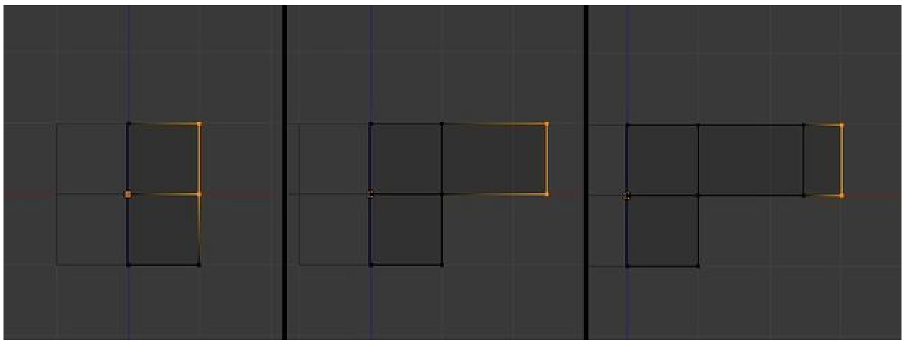
Clipping button.

Finally, with the Clipping button checked, the mirror plane becomes a border that no vertex can cross. Also, when Clipping is active, every vertex that is **on** the mirror sticks to it.

As you can see, the Mirror modifier gives us a lot of features to make our lives easier.

Arms and Legs

Let's create Gus's arms and legs. Using the sequence you just learned, Box Select the two top-right-most vertices (*Extruding the arm in two steps*), which will actually select the other four behind them, for a total of six vertices. Press E to extrude them (or use the Extrude Region button in the Tool Shelf). This will create new movable vertices and faces which you can move with the mouse. Move them one and a half squares to the right, then click LMB to fix their position. Extrude again with E then move the new vertices another half a square to the right. The image below shows this sequence.


Extruding the arm in two steps.

Undo/Redo

Blender has two Undo features, one for Edit Mode and the other for Object Mode.

In Edit Mode press CtrlZ to Undo and keep pressing CtrlZ to roll back changes as long as the Undo buffer will allow; ⇧ ShiftCtrlZ re-does changes. On Macs use ⌘ Cmd instead of Ctrl.

Two things to remember:

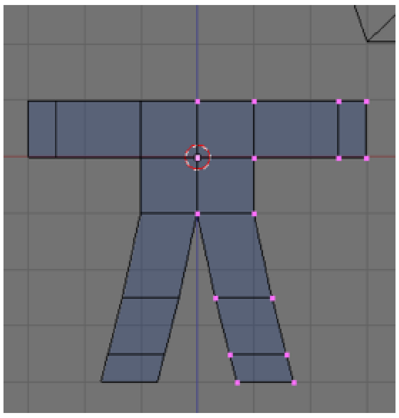
- Undo in Edit Mode works only for the Object currently in that mode.
- Undo data is not lost when you switch out of Edit Mode, but it is as soon as you start editing a *different* Object in Edit Mode.

In Object Mode the same shortcuts apply. CtrlZ to undo, ⇧ ShiftCtrlZ to redo. If you made changes in Edit Mode that are not lost for that Object, they will all be undone in one single shot with CtrlZ when this step has its turn.

If you change your mind in the middle of an action, you can cancel it immediately in many cases and revert to the previous state by pressing Esc or RMB.

Coincident vertices

Extruding works by first creating new vertices and then moving them. If in the process of moving you change your mind and press Esc or RMB to cancel, the new vertices will still be there, on top of the original ones! The simplest way to go back to the state before you started extruding is to Undo (CtrlZ). It is sometimes useful to intentionally create new vertices this way and then move, scale or rotate them by pressing G,S or R.


Body.
Gus should now have a left arm that you modeled (he's facing us) and a right arm that Blender added. We will build the left leg the same way by extruding the lower vertices three times. Try to produce something similar to the *Body* image shown to the right. If you are using *Extrude - Region*, you will have to temporarily turn off the Mirroring modifier by unchecking the X option under Axis, and rechecking it after extruding (otherwise Gus will end up with a skirt rather than pants).

You can free the movement of the extruding vertices by clicking MMB after you have pushed E but before you click LMB. If you do not do this your legs will end up going straight down, rather than down and to the side as pictured in *Body*.

Tip: If you want to position exactly, hold down Ctrl while moving things around.

We're done with mirror modeling. In the next steps we will experiment with other techniques. We need to make the right part of our model *real* since nothing done with modifiers is permanent unless we *apply* the changes. With Gus being in Object Mode (press ⇆ Tab if he's still in Edit Mode), click on the *Apply* button of the Mirror modifier.

The Head

Gus needs a head.

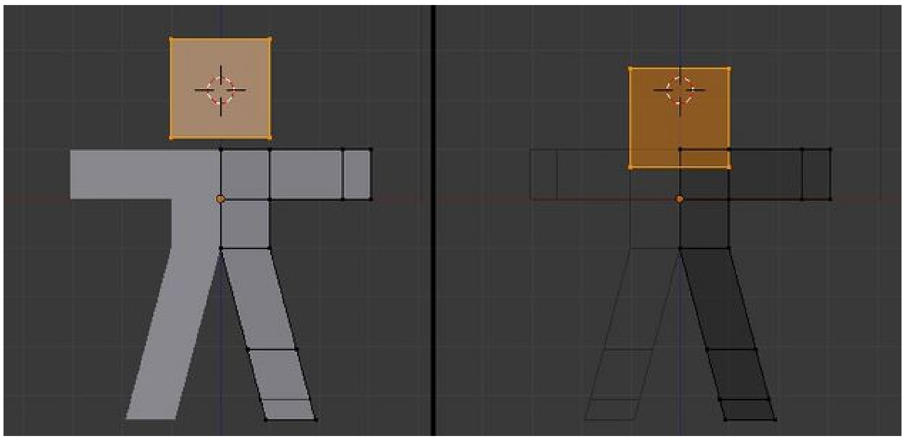
Change back to Edit Mode (press ⇆ Tab)

Move the cursor to exactly one square above Gus's body. To place the cursor at a specific grid point, LMB click to position the cursor near where you want it and then press ⇧ ShiftS to bring up the Snap Menu. Cursor to Grid places the cursor exactly on a grid point. That's what we want right now. Cursor to Selection places it exactly on the selected object, which is sometimes handy.

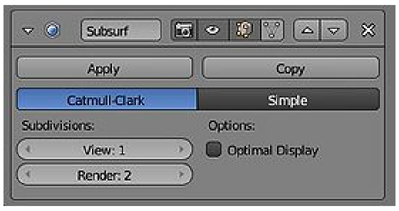
Add a new cube for Gus' head (⇧ ShiftA >> Add >> Cube) (leftmost image of *Adding the head*).

Object Creation
When you add an object while in Edit Mode for another object, the new object becomes part of the existing object. So by adding this cube while we have Gus' body in edit mode, it's automatically part of him.

Now press G to switch to Grab Mode and move the newly created cube down. You can constrain the movement to a straight line by moving the head down a bit and then clicking MMB. Move Gus' new head down about one third of a grid unit then press LMB to fix its position (rightmost image of *Adding the head.*).


Adding the head.

SubSurfaces *(Subsurf)*


Subsurf

Apply Copy

Catmull-Clark Simple

Subdivisions: View: 1 Render: 2

Options: Optimal Display

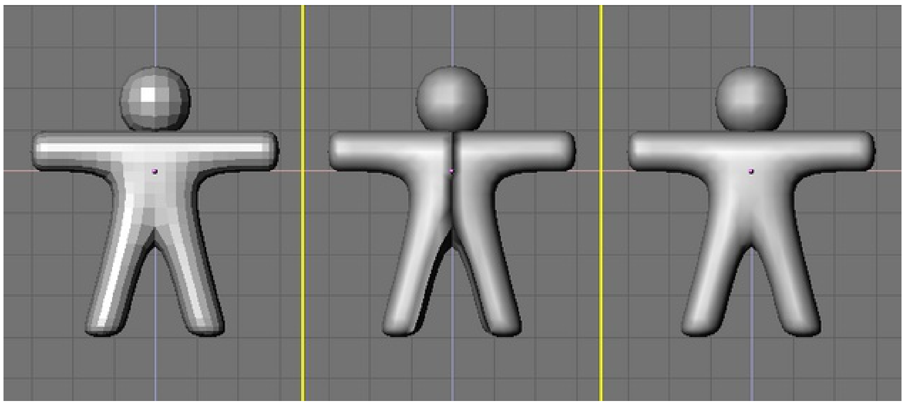
The Subsurf modifier

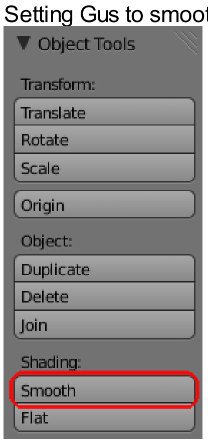
For the next step, we'll need to select all of Gus, and not just his head (use A - maybe twice).

So far what we have produced is a rough figure at best. To make it smoother, locate the Modifier context and add a Subdivision Surface modifier, (*The Subsurf modifier*). Be sure to set both the View and Render NumButtons (located under Subdivisions) to values at or below 2. View sets the level of subdivision you'll see in the 3D viewport; Render sets the level of subdivision used by the renderer.

SubSurfaces
SubSurfacing is an advanced modelling tool that dynamically refines a coarse mesh. It works by creating a much denser mesh and locating the vertices of this finer mesh so that they follow the original coarse mesh smoothly. The shape of the object is still controlled by the location of the coarse mesh vertices, but the rendered shape is a finely smooth mesh.

To have a look at Gus, switch out of Edit Mode (⇆ Tab) and to Solid display mode using Z (in the case it isn't active). He should look like (*Setting Gus to smooth*).


Setting Gus to smooth

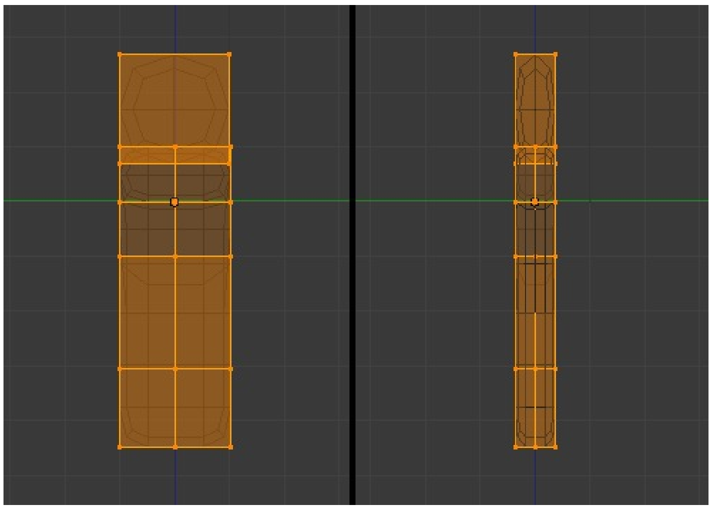


Object Tools.

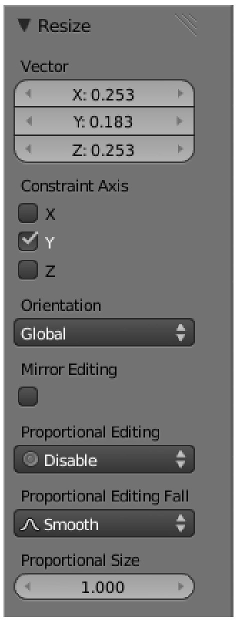
To make smooth Gus look even smoother, press the Smooth button found under Shading in the Tool Shelf of the 3D Window (T). Gus will now appear smooth, although he may wear some funny black lines in his middle. This is usually avoided if you used the Mirror modifier, but it might happen when extruding and flipping as was done before the Mirror modifier was introduced (*Setting Gus to smooth.*, middle). These lines appear because the SubSurf's finer mesh is computed using information about the coarse mesh's normal directions (the direction perpendicular to a face), which may not all point in the right direction (some face normals might be pointing outward and some pointing inward). To reset the normals, switch back to Edit Mode (⇆ Tab), select all vertices (A), and press CtrlN (recalculate vertex normals to point outside). Now Gus should be nice and smooth (*Setting Gus to smooth*, right).

Press MMB and drag the mouse around to view Gus from all angles. Oops, he is too thick!

Constrained Scaling


Slimming Gus using constrained scaling

Let's make Gus thinner:


▼ Resize

Vector

X: 0.253

Y: 0.183

Z: 0.253

Constraint Axis

☐ X

☑ Y

☐ Z

Orientation

Global

Mirror Editing

☐

Proportional Editing

Disable

Proportional Editing Fall

Smooth

Proportional Size

1.000

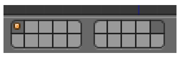
Parameters of the last action in the Tool Shelf

- Switch to Edit Mode if you are not there already (⇆ Tab), then back to Wireframe mode. (Z), Switch to side view using Num3 and select all vertices with A. You can do the following steps just as well in Object Mode, if you like.
- Press S and start to move the mouse horizontally. (Click MMB to constrain scaling to just one axis or press Y to obtain the same result). If you now move the mouse toward Gus he should become thinner but remain the same height.
- The tab *Resize* on the Tool Shelf shows the scaling factor. Press and hold Ctrl in order to change the scale factor in discrete steps of 0.1. Scale Gus down so that the factor is 0.2, then confirm this dimension by clicking LMB . If that last transformation went wrong, you can still change its parameters. They are displayed and editable at the already mentioned place (see *Parameters of the last action in the Tool Shelf*).
- Return to Front view (Num1) and to Solid mode (Z), then rotate your view via MMB . Gus is much better now!

Let's see what Gus looks like

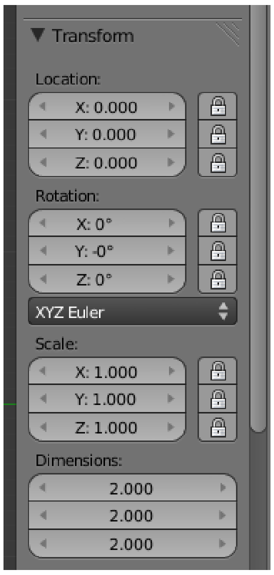
We're just about ready to see our first rendering, but first, we have some work to do.

- Switch to Object Mode if not already there (⇆ Tab).


Making both layer 1 and 10 visible.

- ⇧ Shift LMB on the top right small button of the layer visibility buttons in the 3DWindow toolbar (*Making both layer 1 and 10 visible.*) to make both Layer 1 (Gus's layer) and Layer 10 (the layer with the camera and the lamp) visible.

A Tip
Remember that the last layer selected is the active layer, so all subsequent additions will automatically be on layer 10.



Transform

Location:
X: 0.000
Y: 0.000
Z: 0.000

Rotation:
X: 0°
Y: -0°
Z: 0°

XYZ Euler

Scale:
X: 1.000
Y: 1.000
Z: 1.000

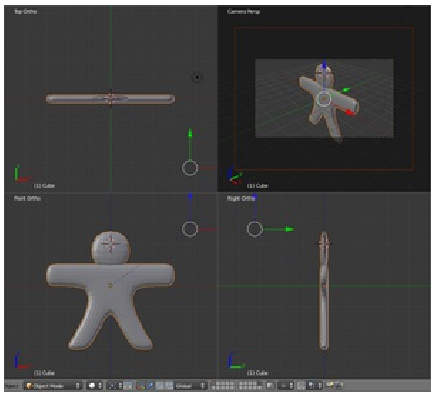
Dimensions:
2.000
2.000
2.000

The Transform Panel

- Press N to bring up the Properties Shelf and find the Transform panel there (*The Transform Panel*). The location is specified by the X,Y,Z values.
- Select the camera (RMB) and move it to a location like (x=7, y=-10, z=7). Do this by pressing G and dragging the camera. You may need to change views and move the camera a second time to adjust all three coordinates. If you prefer to enter numerical values for an object's location you can do so by clicking LMB on a Value and then entering the desired value.

Camera setup

To make the camera point at Gus, keep your camera selected then select Gus via ⇧ Shift RMB . The camera should be dark orange (selected) and Gus light light orange (selected and active). Now press CtrlT and select the TrackTo Constraint entry in the pop up. This will force the camera to track Gus and always point at him. This means that you can move the camera wherever you want and be sure that Gus will always be in the center of the camera's view.



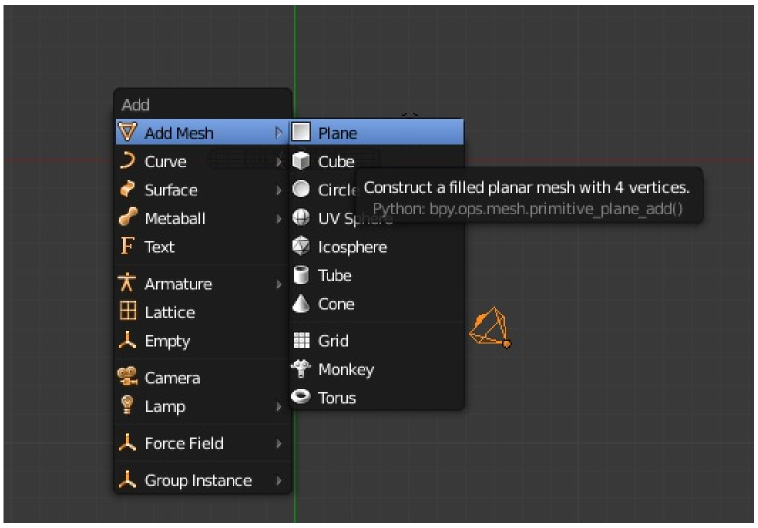
Camera position with respect to Gus.

(*Camera position with respect to Gus*) shows top, front, side and camera view of Gus. To obtain a camera view press 0 NumPad or select View>>Camera. To get a Quad View as shown in the picture, press CtrlAltQ or select View>>Quad View.

The Ground

Now we need to create the ground for Gus to stand on.

- In top view (7 NumPad or View>>Top), and in Object Mode, add a plane
- Add a plane (⇧ ShiftA or >>Add>>Mesh>>Plane).



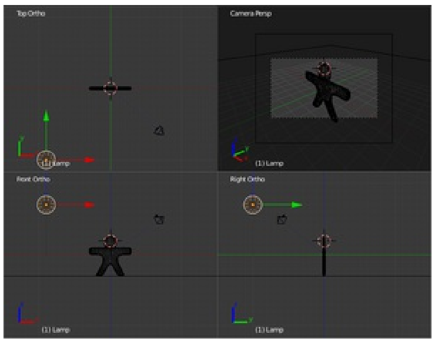
Construct a filled planar mesh with 4 vertices.
Python: bpy.ops.mesh.primitive_plane_add()

Note
It is important to be out of Edit Mode, otherwise the newly added object would be part of the object currently in Edit Mode, as when we added Gus' head.

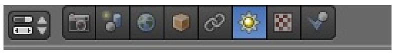
- Switch to Front view (1 NumPad or View>>Front) and move (G) the plane down to Gus's feet, using Ctrl to keep it aligned with Gus.
- Go to Camera view (0 NumPad or View>>Camera) and, with the plane still selected, press S to start scaling.
- Enlarge the plane so that its edges extend beyond the camera viewing area, as indicated by the lighter area in the camera view.

Lights

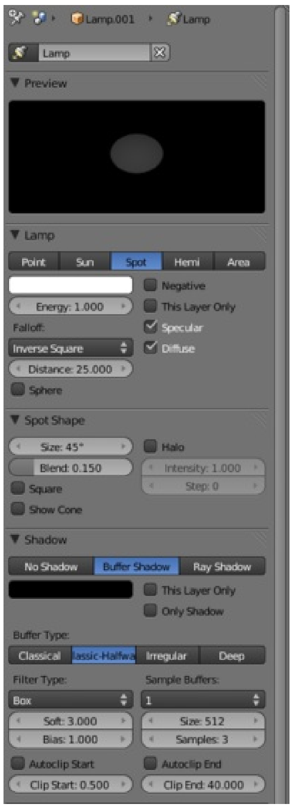
Now, lets add some light!


Inserting a Lamp

- In Top view (7 NumPad), move the existing Lamp light (if you do not have a Lamp light in your scene you can add one with ⇧ ShiftA >>Add>>Lamp>>Lamp) in front of Gus, but on the other side of the camera; for example to (x= -9, y= -10, z=7) (*Inserting a Lamp.*).


The object data button of a lamp

- When the lamp is selected in Object Mode, select the Object Data icon in the properties window (looking like a small sun). You will see a Lamp submenu, with Point, Sun, Spot, Hemi, and Area choices. (*The object data button of a lamp*).

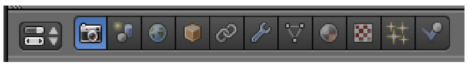

Lamp.001 › Lamp
Lamp
Preview
Lamp
Point Sun Spot Hemi Area
Negative
Energy: 1.000
This Layer Only
Falloff:
Specular
Inverse Square
Diffuse
Distance: 25.000
Sphere
Spot Shape
Size: 45°
Halo
Blend: 0.150
Intensity: 1.000
Step: 0
Square
Show Cone
Shadow
No Shadow Buffer Shadow Ray Shadow
This Layer Only
Only Shadow
Buffer Type:
Classical lassic-Halfwa Irregular Deep
Filter Type: Sample Buffers:
Box 1
Soft: 3.000 Size: 512
Bias: 1.000 Samples: 3
Autoclip Start Autoclip End
Clip Start: 0.500 Clip End: 40.000

The Spot light settings

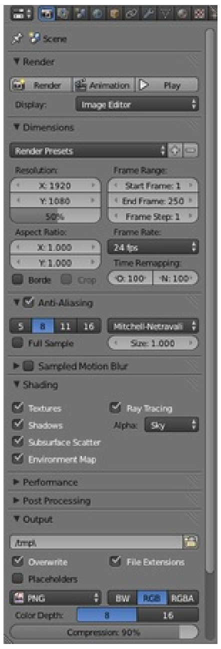
- In the Properties Window Lamp Panel, press the Spot toggle button to make the lamp a Spotlight (*The Spot light settings.*) of pale yellow (R=1, G=1, B=0.9) by clicking on the white button, which is actually a color picker. Adjust Size under Spot Shape to about 40 and Blend: to 1.0.
- Make this spotlight track Gus just as you did for the camera by selecting Spot, ⇧ Shift, then Gus, then by pressing CtrlT>>TrackTo Constraint.
- Add a second lamp that provides more uniform fill light (⇧ ShiftA >>Add>>Lamp>>Hemi). Set its Energy to 0.2 (*Hemi lamp settings*). Move it a little above the camera (x= 7, y= -10, z=9) and set it to track Gus as before.

Two lamps?
Use two or more lamps to help produce soft, realistic lighting, because in reality natural light never comes from a single point.

Rendering


The Render context button

We're almost ready to render. As a first step, press the Render context button in the Properties window header (*The Render context button*).


Scene
Render
Render | Animation | Play
Display: Image Editor
Dimensions
Render Presets
Resolution: X: 1920, Y: 1080, 50%
Frame Range: Start Frame: 1, End Frame: 250, Frame Step: 1
Aspect Ratio: X: 1.000, Y: 1.000
Frame Rate: 24 fps
Time Remapping: O: 100, N: 100
Borde | Crop
✓ Anti-Aliasing
5 | 8 | 11 | 16 — Mitchell-Netravali
Full Sample — Size: 1.000
Sampled Motion Blur
Shading
✓ Textures ✓ Ray Tracing
✓ Shadows Alpha: Sky
✓ Subsurface Scatter
✓ Environment Map
Performance
Post Processing
Output
/tmp\
✓ Overwrite ✓ File Extensions
Placeholders
PNG — BW | RGB | RGBA
Color Depth: 8 | 16
Compression: 90%

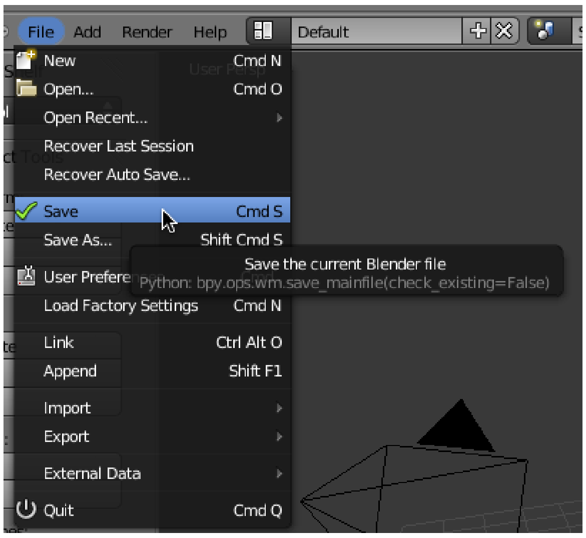
The Render context

We will use the default rendering settings, as shown in (*The Render context*).

Now press the Image button or F12. The result, shown in (*Your first rendering. Congratulations!*), is actually quite poor. We still need materials, and lots of details, such as eyes, and so on. When you're done looking, press F11 to hide the render view.

Your first rendering. Congratulations!

Saving our work



Save the current Blender file
Python: bpy.ops.wm.save_mainfile(check_existing=False)

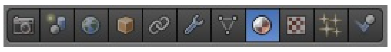
The Save menu

If you have not done so already, now would be a good time to save your work, via the File>>Save menu shown in *The Save menu.' or CtrlS. Blender will warn you if you try to overwrite an existing file.*

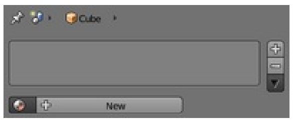
Blender does automatic saves into your system's temporary directory. By default, this happens every five minutes and the file name is a number. Loading these saves is another way to undo unwanted changes.

Materials and Textures

It's time to give Gus some nice cookie-like material.

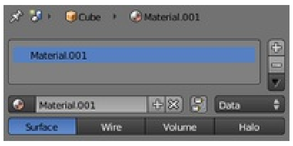

The Material context Button.

- Select Gus. Then, in the Properties Window header, select the Materials button (*The Material context Button.*) to access the Material panels.



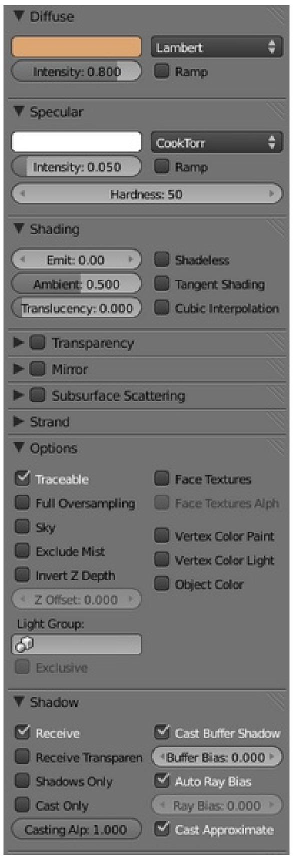
The (almost) empty Material context

- The Properties window will be almost empty because Gus has no materials yet. To add a material, click on the + *New*button in the Material Panel (*The (almost) empty Material context*).



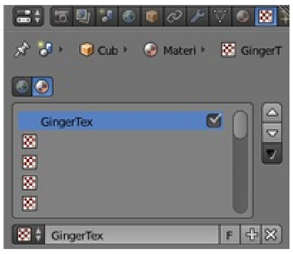
Top of the filled material context

- The Properties window will then be populated by Panels and Buttons. A string holding the Material name, generally "Material.001", will appear in the list box as well as in the Unique Datablock ID box. Click the name in the latter (in the lower part of Top of the filled material contextand change it to something meaningful, like "GingerBread" (don't type the quotes).


Diffuse
Lambert
Intensity: 0.800 Ramp
Specular
CookTorr
Intensity: 0.050 Ramp
Hardness: 50
Shading
Emit: 0.00 Shadeless
Ambient: 0.500 Tangent Shading
Translucency: 0.000 Cubic Interpolation
Transparency
Mirror
Subsurface Scattering
Strand
Options
✓ Traceable Face Textures
Full Oversampling Face Textures Alph
Sky Vertex Color Paint
Exclude Mist Vertex Color Light
Invert Z Depth Object Color
Z Offset: 0.000
Light Group:
Exclusive
Shadow
✓ Receive ✓ Cast Buffer Shadow
Receive Transparen Buffer Bias: 0.000
Shadows Only ✓ Auto Ray Bias
Cast Only Ray Bias: 0.000
Casting Alp: 1.000 ✓ Cast Approximate

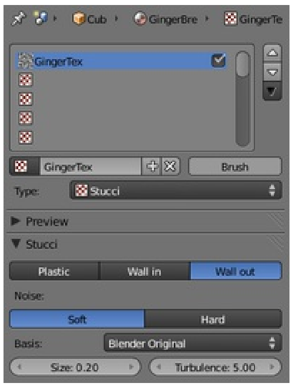
A first gingerbread material.

- Modify the default values as per (*A first gingerbread material*) to obtain a first rough material. Note that you may have to expand some panels by clicking LMB on the small triangle besides their header.



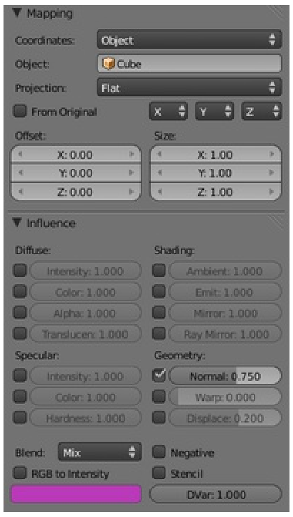
The Texture context button

- Press the Texture context Button in the Properties window header (*The Texture context button*) and select Add new. We're adding a texture in the first channel. Call it "GingerTex."


Cub GingerBre GingerTe
GingerTex
GingerTex Brush
Type: Stucci
Preview
Stucci
Plastic Wall in [✓ Wall out]
Noise:
[✓ Soft] Hard
Basis: Blender Original
Size: 0.20 Turbulence: 5.00

The Texture context

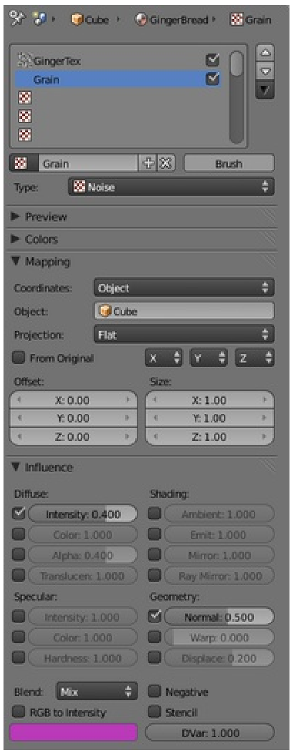
- Change the Type from Clouds to Stucci and set all parameters as in (*The Texture context*).



```
▼ Mapping
Coordinates: Object
Object: Cube
Projection: Flat
From Original   X   Y   Z
Offset:            Size:
X: 0.00            X: 1.00
Y: 0.00            Y: 1.00
Z: 0.00            Z: 1.00
▼ Influence
Diffuse:                 Shading:
Intensity: 1.000         Ambient: 1.000
Color: 1.000             Emit: 1.000
Alpha: 1.000             Mirror: 1.000
Translucen: 1.000        Ray Mirror: 1.000
Specular:                Geometry:
Intensity: 1.000       ✓ Normal: 0.750
Color: 1.000             Warp: 0.000
Hardness: 1.000          Displace: 0.200
Blend: Mix               Negative
RGB to Intensity         Stencil
                         DVar: 1.000
```

Settings for the Stucci texture

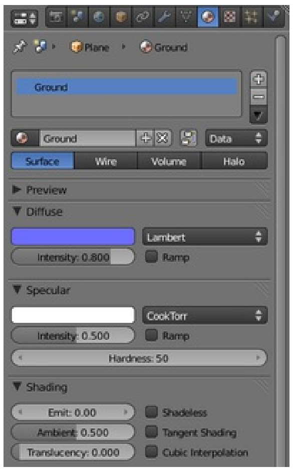
- Set the Mapping and Influence panel of the Texture context as in (*Settings for the Stucci texture*): Deselect the Color Checkbox and set the Normal Checkbox, then change the Normal slider to **0.75**. These changes will make our Stucci texture act as a "bumpmap" and make Gus look more biscuit-like.



```
Cube ▸ GingerBread ▸ Grain
GingerTex ✓
Grain ✓
Grain                    Brush
Type: Noise
▶ Preview
▶ Colors
▼ Mapping
Coordinates: Object
Object: Cube
Projection: Flat
From Original   X   Y   Z
Offset:            Size:
X: 0.00            X: 1.00
Y: 0.00            Y: 1.00
Z: 0.00            Z: 1.00
▼ Influence
Diffuse:                 Shading:
✓ Intensity: 0.400       Ambient: 1.000
Color: 1.000             Emit: 1.000
Alpha: 0.400             Mirror: 1.000
Translucen: 1.000        Ray Mirror: 1.000
Specular:                Geometry:
Intensity: 1.000       ✓ Normal: 0.500
Color: 1.000             Warp: 0.000
Hardness: 1.000          Displace: 0.200
Blend: Mix               Negative
RGB to Intensity         Stencil
                         DVar: 1.000
```

Settings for an additional Noise texture

- Now select the second line in the Texture list of the Texture context and add a second texture. Name it "Grain", and adjust the settings to match (*Settings for an additional Noise texture*). The texture itself is a plain Noise texture.



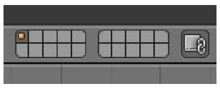
Ground
Data
Diffuse: Lambert
Intensity: 0.800
Ramp
Specular: CookTorr
Intensity: 0.500
Ramp
Hardness: 50
Shading
Emit: 0.00
Shadeless
Ambient: 0.500
Tangent Shading
Translucency: 0.000
Cubic Interpolation

A very simple ground material

- Give the ground an appropriate material, such as the dark blue one shown in (*A very simple ground material*). Feel free to choose your favorite shade of blue.

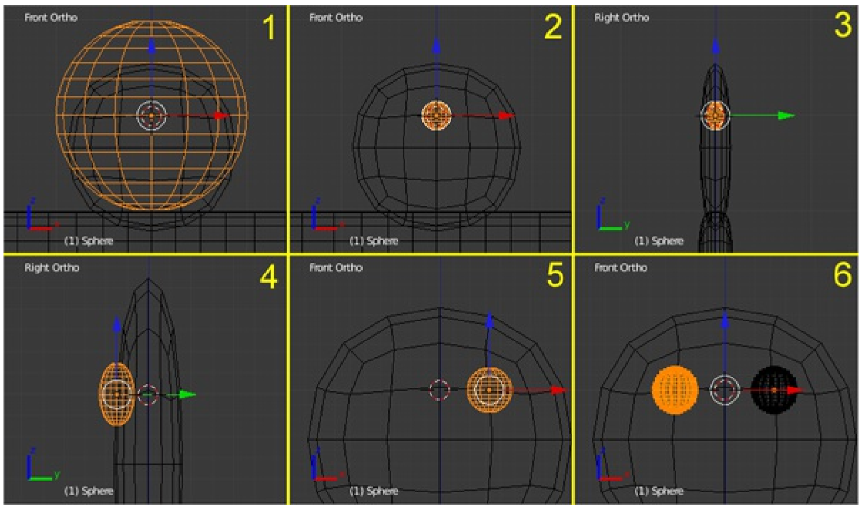
Eyes and detail

To give some finishing touches we'll add eyes and some other details.


Layer visibility

- First make Layer 1 the only one visible by clicking with LMB on the layer 1 button (*Layer visibility*). This will hide the lamps, camera, and ground.

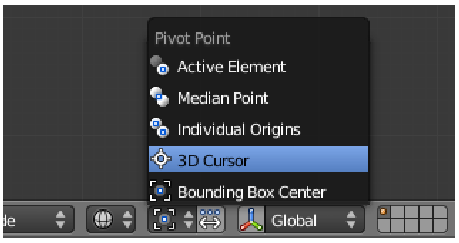
- Place the cursor at the center of Gus's head. (Remember that you are in 3D so be sure to check at least two views to be sure!)
- In Object Mode, add a sphere (⇧ ShiftA >>ADD>>Mesh>>UVsphere). Press F6 and change the number of segments (meridians) to *16*. You can see the result under *1* in *Creation of the eyes*.
- Scale the sphere down (S) to a factor of about **0.15** in all dimensions, then switch to side view (3 NumPad) and scale it only in the horizontal direction (Y) a further **0.5** (see the images 2 and 3 in *Sequence for creation of the eyes*).



- Zoom a little if necessary via + NumPad, Wheel, or Ctrl MMB, and drag the sphere (G) to the left so that it is halfway into the head, as shown in image 4 of *Creation of the eyes*.
- Return to front view (1 NumPad) and move the sphere sideways, to the right. Place it where Gus should have an eye (picture 5 in *Creation of the eyes*).

Flipping a duplicate around the cursor

- Switch to Edit Mode (⇆ Tab). Select the crosshair pivot button (*pivot: 3D Cursor*)in the header of the 3D window (the 3D Transforms Manipulator jumps from the sphere to the cursor). All vertices of the eye should be selected (if not, press A to select all), now press ⇧ ShiftD or use Duplicate in the Tool Shelf to duplicate and Esc to stop placing the copy with the mouse.



Pivot: 3D Cursor

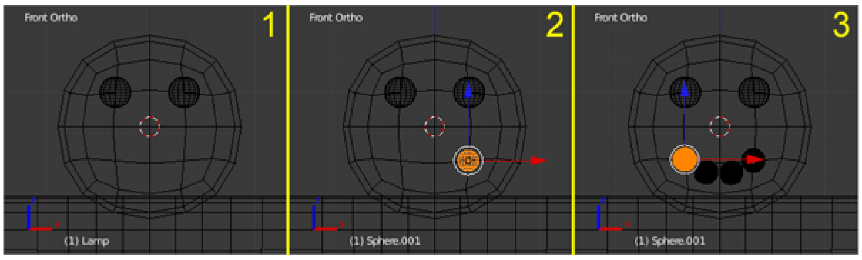
- Press CtrlM to mirror, X to mirror around the X axis, followed by LMB or ↵ Enter to confirm the mirror. Return the pivot button to its default setting (Median Point). The result can be seen in picture 6 of *Creation of the eyes*.

Mirroring
Mirroring is also possible in object mode using CtrlM.

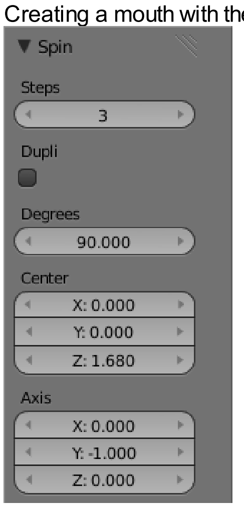
Now Gus has two eyes.

Mouth

- Exit Edit Mode (⇆ Tab), and place the cursor as close as you can (remember the ⇧ ShiftS key) to the center of Gus's face. Add a new sphere and scale and move it exactly as done before for the eyes, except make its allover scale smaller (0.1 instead of 0.15). Place it below and to the right of the cursor, centered on the SubSurfed mesh vertex, as shown in picture 2 of *Creating a mouth with the Spin tool*.



Creating a mouth with the Spin tool

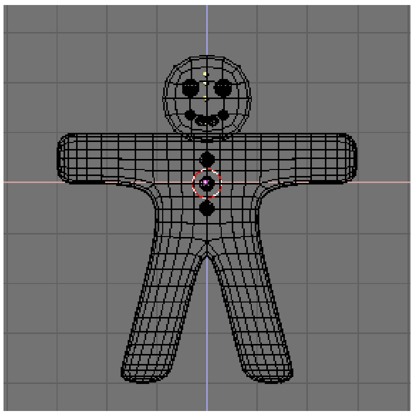


▼ Spin
Steps: 3
Dupli
Degrees: 90.000
Center
X: 0.000
Y: 0.000
Z: 1.680
Axis
X: 0.000
Y: -1.000
Z: 0.000

The Spin Tools options on the Tool Shelf

- Switch to Edit Mode (⇆ Tab). Now use AltR or LMB on Spin on the Tool Shelf. Several copies of the sphere appear.
- On the lower Tool Shelf, (Press F6 if not visible), set the details of the Spin: Set Degr: to **90**, Steps: to **3**. The result should be Gus's mouth, like image 3 in *Creating a mouth with the Spin tool*.

(EDIT REMARK: The properties of the last command (as *Spin* in the above descriptions) don't show in every 3D window (at least in the editors Version of Blender (V2.5 beta, Mac)) so don't panic - it is there somewhere (at least in the default screen))


The complete Gus!

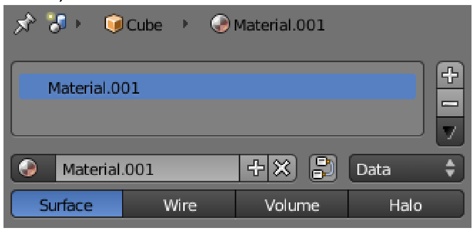
- Now go back to Object Mode and add three more spheres (below the head and centered along the Z-axis) to form Gus's buttons. Once you have made one button, you can simply exit Edit Mode, press ⇧ ShiftD to create a duplicate, and move the duplicate into place, as shown in *The complete Gus!*.

Attaching the spheres
If we want to be able to grab Gus and move him around as a whole (this goes beyond the animation in the second part of this tutorial), we now need to attach the small spheres representing eyes, mouth, and buttons to the body. Enter Object Mode and press A until nothing is selected. Now right click one sphere (if more than one is selected as a group, that's ok). Holding ⇧ Shift, select the body. Then hit CtrlP and left click Object on the pop up. Deselect everything and repeat the process to attach each element.

Eyes material

Give the eyes a chocolate-like material. Give the mouth a white sugar-like material, and Gus's buttons could use a bit of color too. Remember what you did when giving a material to Gus himself and be a bit creative; soon Gus will show top-notch gingerbread fashion.

Objects sharing a material
To give one object the same material as another object (i.e. reusing the white material of the mouth on one of the buttons), select that material in the **Material Menu** list which appears when you press the button Browse ID Data besides the Data Block ID Name in the Material context of the Properties window (In *Material context* the checkered little sphere shown besides *Material.001* is what you need).



Rendering

Once you have finished assigning materials, make layer **10** visible again (remember how? Hint, look at the 3D window header), so that lights and the camera also appear, and do a new rendering (F12).

The result should look more or less like (*The complete Gus still rendering*).

The complete Gus still rendering.

Saving

Save your image by using F3 in the UV/Image Editor that is showing the render result. Enter the name of your image in the file window, choose a destination and save.

You can choose the image format (JPEG, PNG, and so on) by setting it in the shelf to the right of the File Browser.

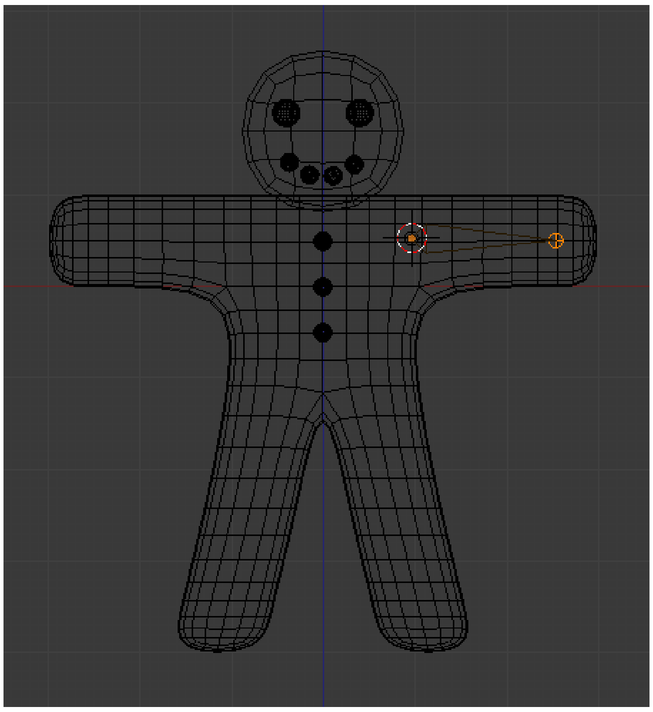
Blender adds an extension to the file name; as you are probably used to (this is new in V2.5, in V2.4x you had to do it manually).

Your First Animation in 30 plus 30 Minutes Part II

If all we needed was a still picture, our work would be done, but we want Gus to move. The next step is creating a skeleton or armature, which will allow us to move him. This is called rigging. Gus will have very simple rig: four limbs (two hands and two legs) and several joints (no elbows, only knees), no feet and no arms.

Rigging

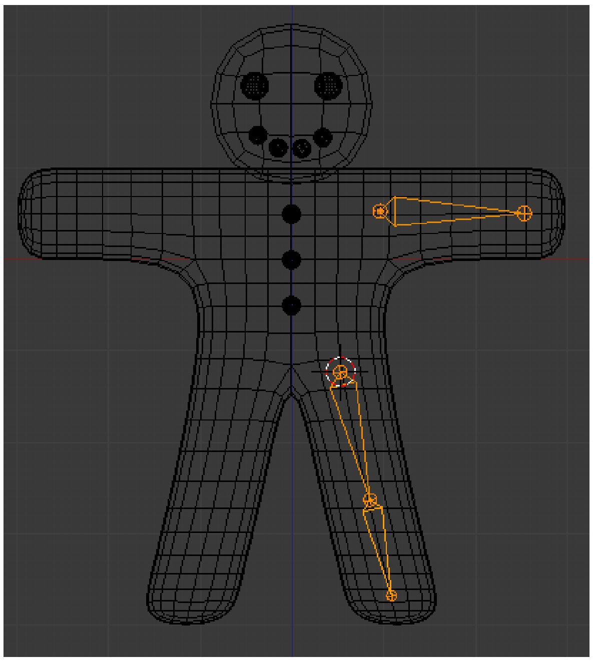
To add rigging:


Adding the first bone - hand to elbow.

- In Object mode, set your 3D cursor where Gus's shoulder is, and press ⇧ ShiftA >> Add >> Armature >> Single bone. A rhomboidal object will appear, which is a bone of the armature system. Enter Edit mode. The end, or *tip*, of the bone is selected (yellow).
- Now in Edit mode, place the tip of the bone in Gus's hand by grabbing (G) and moving it, (*Adding the first bone, an elbowless arm*). We don't need any other bones right now. You should now have one bone running from the shoulder to the hand area. As you move the tip, you will notice that the whole bone gets bigger – you really are scaling up the bone.

To make the process more successful, please periodically look at the Gingerbread man and armature from many different viewpoints to make sure the armature is *inside* the gingerbread man, just as bones are inside a human body. Skinning will fail if the bones are for example in front or in back of the body. Inspection from many different viewpoints is a common 3D model creation technique.

About Bones' Ends
Bones' ends can have different names. In Blender, currently they are called "head"/"tail" (the first being the "large" end, and the second the "thin" end). However, historically, they have been named "root"/"tip", which is often considered somewhat less confusing…

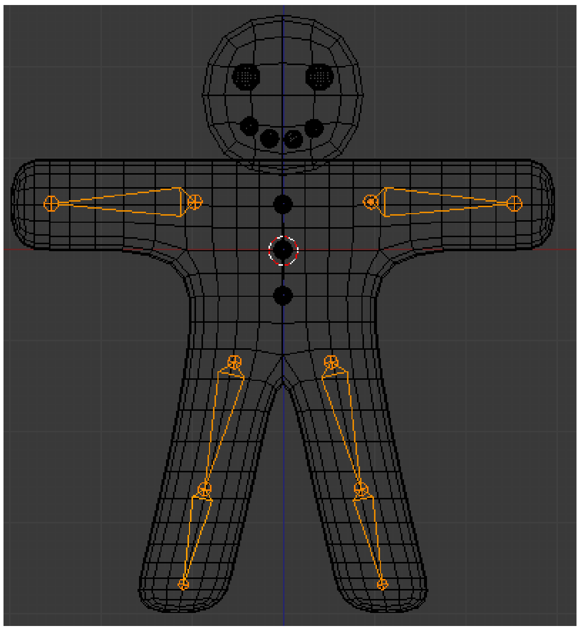

Adding the second and third bones, a leg bone chain.

- Stay in Edit mode, then move the cursor to where the hip joint will be and add a new bone ⇧ ShiftA.
- Grab (G) and move the yellow tip of the new bone to the knee area.
- Now “chain” a new bone from the knee to the foot by Ctrl LMB clicking in the area of the foot. A new *chained* bone will appear automatically linked with the knee and ending at the foot, (*Adding the second and third bones, a leg bone chain*). Another way of chaining the new bone would be to extrude using the E shortcut. This variation creates the new bone and places you in grab mode automatically. Click LMB to validate the current bone’s tip position.

Bone position
The bones we are adding will deform Gus’s body mesh. To produce a neat result, try to place the bone joints as shown in the illustrations.

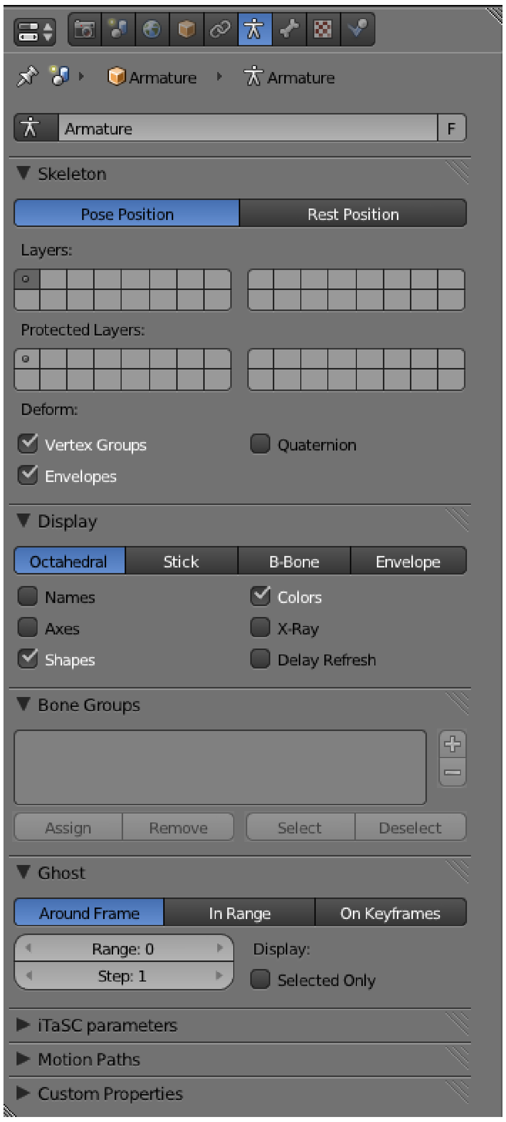
Bone roll
To get the bones lined up as in (*Adding the second and third bones, a leg bone chain*) you may need to adjust the Bone Roll by pressing CtrlN, 3 with the lower leg bone selected.

We now have three bones that make up Gus’s armature.


The complete armature after duplicating and flipping.

- Now place the cursor in the center (⇧ ShiftC) and select all bones with A. Duplicate them with ⇧ ShiftD and immediately exit grab mode with Esc. Make sure the cursor is selected as the rotation/scaling pivot (Pivot drop-down list in the 3D window header). Flip the duplicated bones along the X axis relative to the cursor with CtrlM and then X. Click LMB to confirm the mirror operation. You end up with (*The complete armature after duplicating and flipping*).

When any bone is selected the Object Data context shows settings for the Armature as a whole, such as the name of the Armature Object and settings for displaying the armature, while the Bone context shows the name of the active Bone and bone specific settings.


Armature › Armature

Armature F

Skeleton

[✓ Pose Position] Rest Position

Layers:

Protected Layers:

Deform:

✓ Vertex Groups
Quaternion
✓ Envelopes

Display

[✓ Octahedral] Stick B-Bone Envelope

Names
✓ Colors
Axes
X-Ray
✓ Shapes
Delay Refresh

Bone Groups

Assign Remove Select Deselect

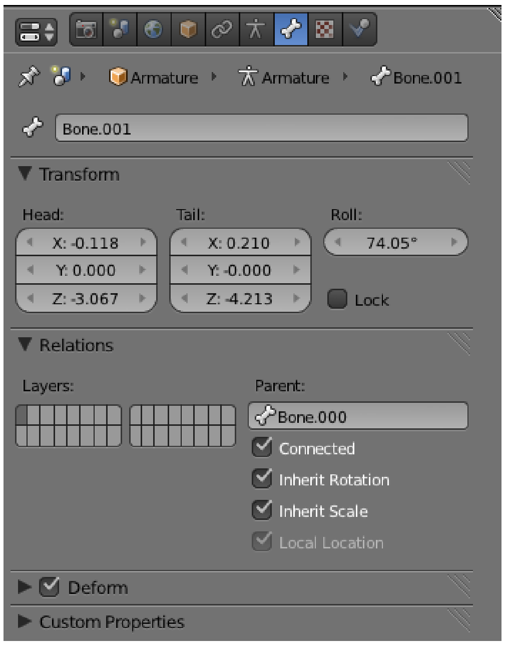
Ghost

[✓ Around Frame] In Range On Keyframes

Range: 0
Step: 1
Display:
Selected Only

iTaSC parameters
Motion Paths
Custom Properties

Armature context.


Armature › Armature › Bone.001

Bone.001

Transform

Head: X: -0.118 Y: 0.000 Z: -3.067
Tail: X: 0.210 Y: -0.000 Z: -4.213
Roll: 74.05°
Lock

Relations

Layers:
Parent: Bone.000
✓ Connected
✓ Inherit Rotation
✓ Inherit Scale
✓ Local Location

✓ Deform
Custom Properties

Armature Bones context.

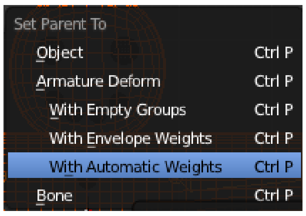
Check the Names checkbox (Armature context, Display panel) to see the names of the bones in 3D views, then select each bone and LMB click on the name of the bone in the Bones context to change the bone names to something appropriate like `Arm.R`, `Arm.L`, `UpLeg.R`, `LoLeg.R`, `UpLeg.L` and `LoLeg.L`, see (*Armature Bones context*). Exit Edit mode with ⇆ Tab.

Naming Bones
It is very important to name your bones with a trailing "`.L`" or "`.R`" to distinguish between left and right ones, so that the Action Editor will be able to automatically flip your poses.

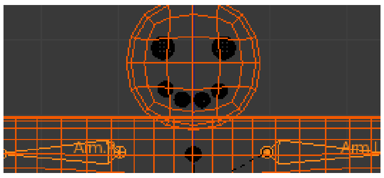
Skinning

Now we must make it such that a deformation in the armature causes a matching deformation in the body. We do this with *skinning*, which assigns vertices to bones so that the formers are subject to the latter's movements.

- In Object mode, select Gus's body, then ⇧ Shift select the armature so that the body is dark orange and the armature is light orange.
- Now we need to parent the body to the armature. That is achieved by pressing CtrlP). The (*Parenting menu*) will appear. Select the Armature Deform >> With Automatic Weights entry.



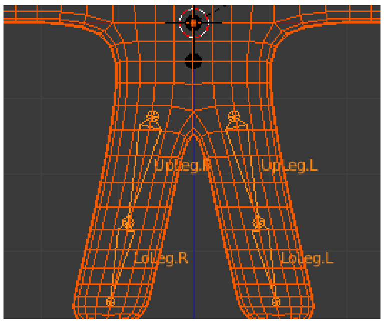
Parenting menu.



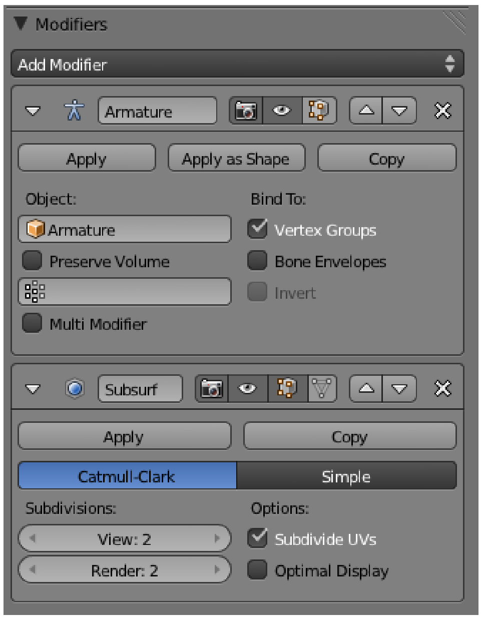
Vertex Groups, Envelopes and the order of modifiers
When skinning a mesh, the mesh object will get an Armature modifier attached to the bottom of its modifier stack. Gus' mesh already has a Subdivision Surface modifier on the stack. In order to get a smooth deformation of the mesh you should move the Armature modifier to a position above the Subdivision Surface modifier. You do this by clicking the Move Modifier buttons in the Modifier context (*The modifier stack in the Modifiers context*). Also, each bone has an area of influence called Envelope. The armature will deform the mesh from both the assigned vertex groups and the bone envelopes. This may lead to unwanted results, so in our case it is important to disable Bone Envelopes in the Armature Modifier (*The modifier stack*

in the Modifiers context).

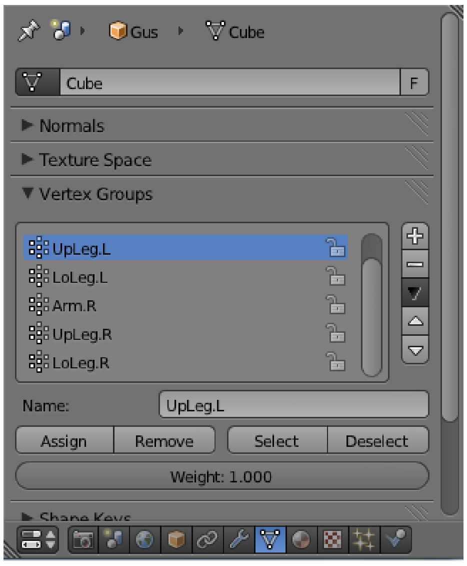
- Now select just Gus's body and switch to Edit mode (⇆ Tab). Notice in the Object Data context the presence of the Vertex Groups group of controls in the Vertex Groups panel (*The vertex groups controls in the Object Data context*).



Armature parented.

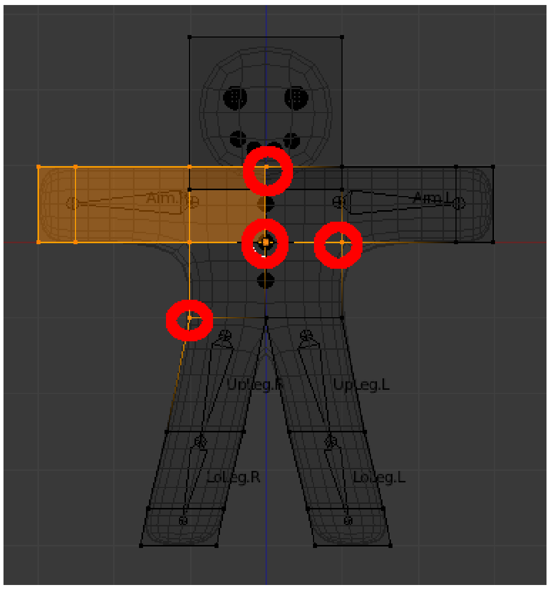


The modifier stack in the Modifiers context.



The vertex groups controls in the Object Data context.

By scrolling in the Vertex Group panel, you can see all available vertex groups – six in our case. But a truly complex character, with hands and feet completely rigged, will have tens of them! See (*The vertex groups controls in the Object Data context*). The buttons Select and Deselect (de)select all vertices of the current selected group, which allows you to see which vertices belong to which group.



Gus in Edit mode with all the vertices of group `Arm.R` selected.

Select the right arm group (`Arm.R`) and, with all vertices deselected (A, if needed), press Select. You should see something like (*Gus in Edit mode with all the vertices of group `Arm.R` selected*).

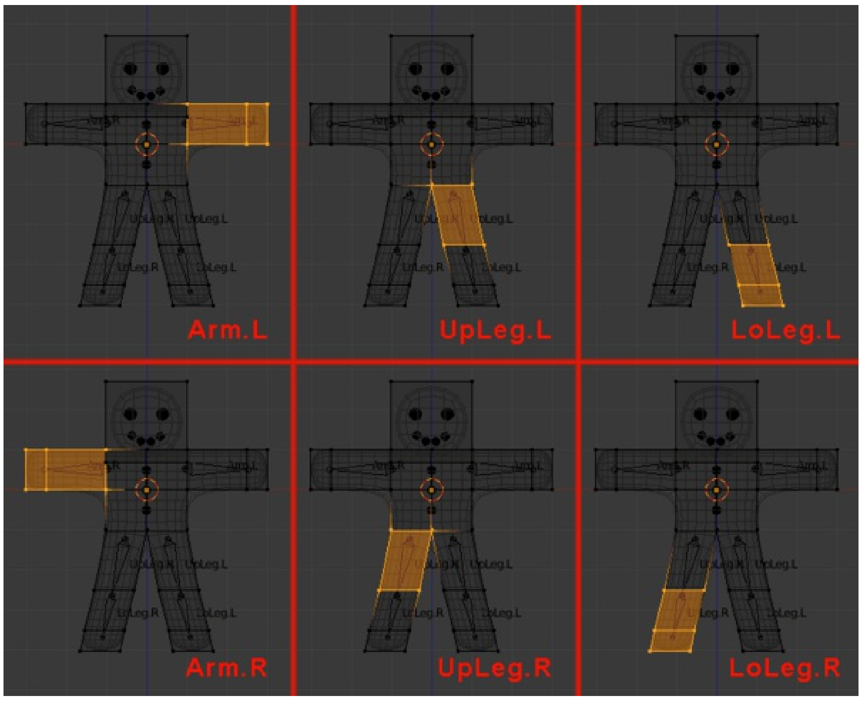
If you don't see the same thing then you probably placed the bones in just the right place such that the *auto skinning* process did a better job of matching vertices with bones. It is highly unlikely that the skinning process matched the vertices to the bones as exactly as you may expect. This requires that you begin to manually adjust the grouping as described in the following sections.

The vertices marked with red circles in (*Gus in Edit mode with all the vertices of group `Arm.R` selected*) belong to the deformation group, however, they should not.

The *auto skinning* process found that they were very close to the bone so it added them to the deformation group. We don't want them in this group since some are on the opposite side of Gus and some are in the chest, adding them to the deformation group would deform those body parts as well.

To remove them from the group, deselect all the other vertices, those which should *remain* in the group, using box selection (B), but with MMB, not LMB, to define the box, so that all vertices within the box become deselected.

Once only the "undesired" vertices are selected, press the Remove button (*The vertex groups controls in the Object Data context*), to eliminate them from the group `Arm.R`. Deselect all (A) then check another group. Check them all and be sure that they look like those in (*The six vertex groups*).



The six vertex groups.

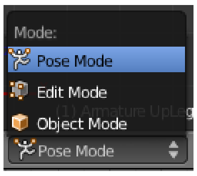
Vertex groups

Be very careful when assigning or removing vertices from vertex groups. If later on you see unexpected deformations, you might have forgotten some vertices, or placed too many in the group. You can modify your vertex groups at any time.

Other details
Our deformations will affect only Gus's body, not his eyes, mouth, or buttons, which are separate objects. While this is not an issue to consider in this simple animation, it's one that must be taken into account for more complex projects, for example by parenting (to vertices) or otherwise joining the various parts to the body to make a single mesh (all these options are detailed in the manual).

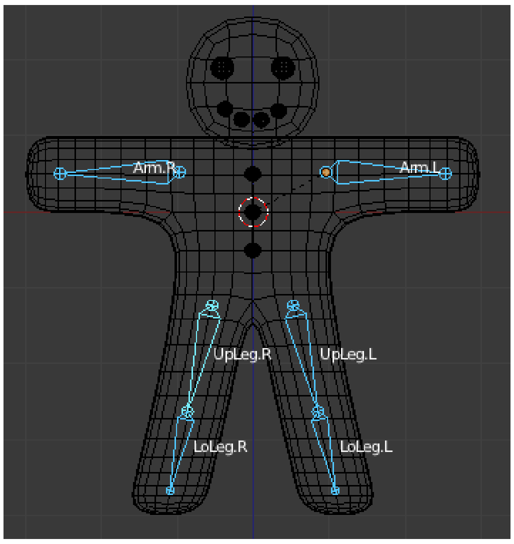
Posing

Once you have a rigged and skinned Gus you can start playing with him as if he were a doll, moving his bones and viewing the results.



Mode menu in the 3D window header.

- Select the armature only, then select Pose Mode from the Mode menu (*Mode menu in the 3D windowheader*) – or simply hit Ctrl⇆ Tab. This option is only available when an armature is selected.
- The selected bones in the armature will turn blue. You are now in Pose mode. If you now select a bone, and you move it (G), or rotate it (R), the body will deform accordingly!



You are in Pose mode now!

Original position

Blender remembers the original position of the bones. You can set your armature back by pressing AltR to clear the bones’ rotation, and AltG to clear their location. Alternatively, the Rest Position button in the Object Data context may be used to temporarily show the original position.

Inverse Kinematics

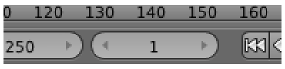
Inverse Kinematics (IK) is where you actually define the position of the *last* bone in the chain, often called an “*end effector*”. All the other bones assume an algorithmic position, automatically computed by the *IK solver*, to keep the chain without gaps (i.e. IK will mathematically solve the chain positions for us). This allows a much easier and precise positioning of hands and feet using IK.

Forward Kinematics

While handling bones in Pose mode, notice that they act as rigid, inextensible bodies with spherical joints at the end. You can grab only the first bone of a chain and all the others will follow it. All subsequent bones in the chain cannot be grabbed and moved, you can only rotate them, so that the selected bone rotates with respect to the previous bone in the chain while all the subsequent bones of the chain follow its rotation.

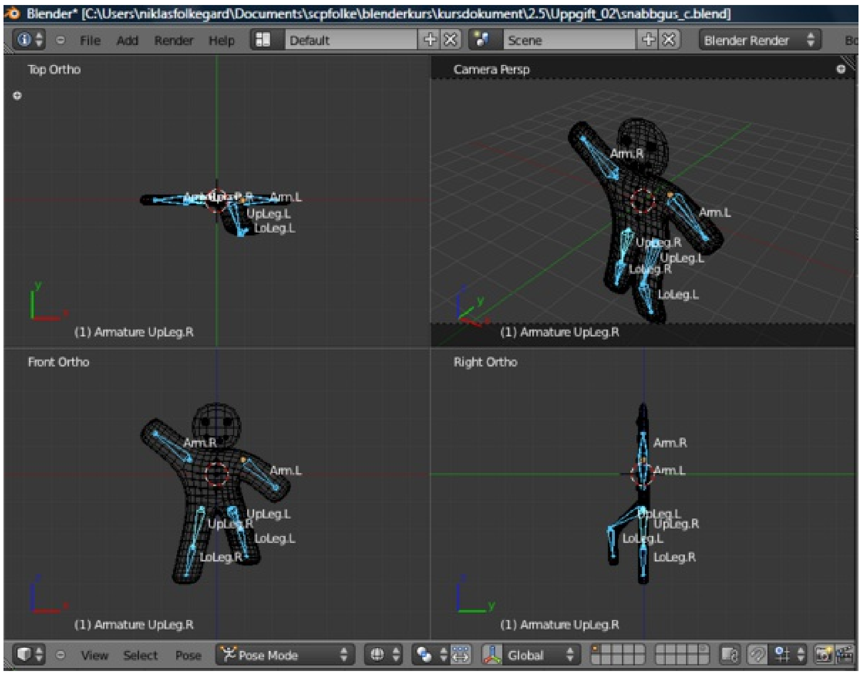
This procedure, called *Forward Kinematics* (FK), is easy to follow but it makes precise location of the last bone in the chain difficult.

We’ll make Gus walk, using FK, by defining four different poses relative to four different stages of a stride. Blender will do the work of creating a fluid animation.

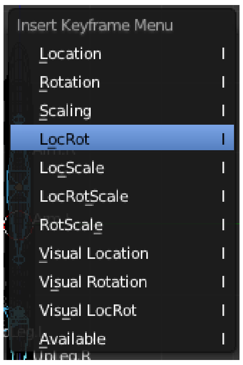


The current frame numeric field in the Timeline window header.

- First, verify that you are at frame **1** of the timeline. The frame number appears in a numeric field in the Timeline window header (*The current frame numeric field in the Timeline windowheader*). If it is not set to **1**, set it to **1** now.
- Now, by rotating only one bone at a time (R), we’ll raise `UpLeg.L` and bend `LoLeg.L` backwards while raising `Arm.R` a little and lowering `Arm.L` a little, as shown in (*Our first pose*).

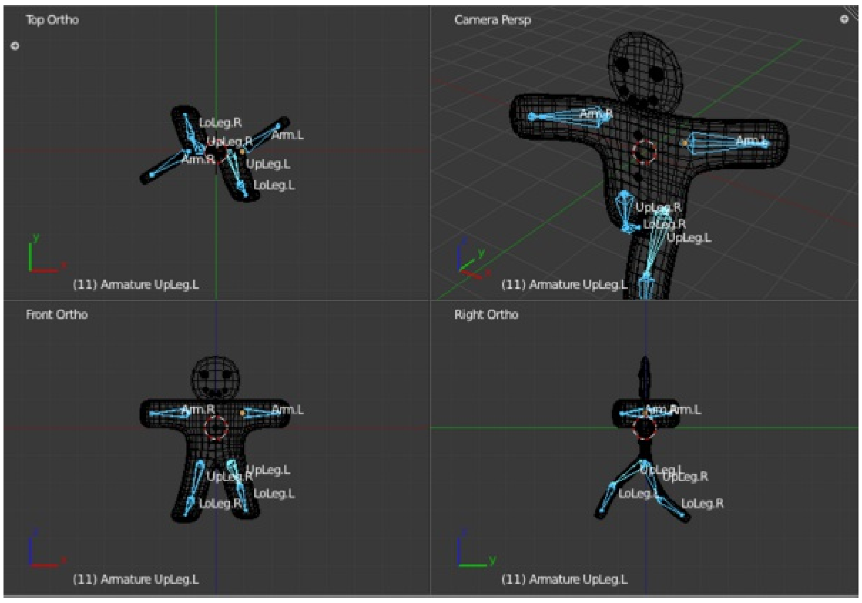

Blender* [C:\Users\niklasfolkegard\Documents\scpfolke\blenderkurs\kursdokument\2.5\Uppgift_02\snabbgus_c.blend]
File Add Render Help Default Scene Blender Render
Top Ortho
Camera Persp
Front Ortho
Right Ortho
Arm.R
Arm.L
UpLeg.L
LoLeg.L
UpLeg.R
LoLeg.R
(1) Armature UpLeg.R
View Select Pose Pose Mode Global

Our first pose.

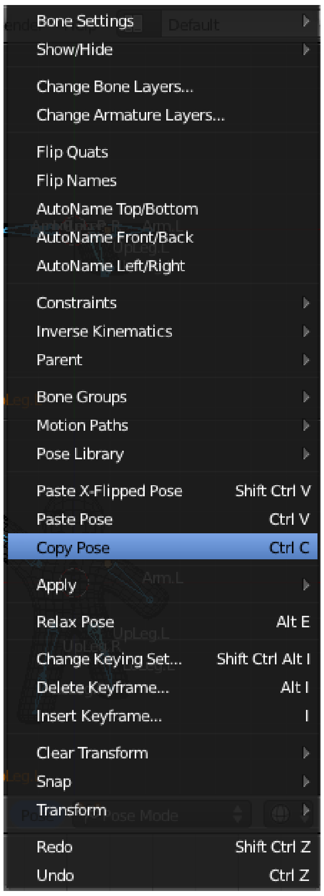


Storing the pose to the frame.

- Select all bones with A. With the mouse pointer on the 3D window, press I. A menu pops up (*Storing the pose to the frame*). Select LocRot from this menu. This will get the position and orientation of all bones and store them as a pose at frame **1**. This pose represents Gus in the middle of his stride, while moving his left leg forward and above the ground.
- Now move to frame **11** either by entering the number in the numeric field or by pressing ↑. Then move Gus to a different position, like (*Our second pose*). Start with clearing the rotation on both arms using AltR as mentioned earlier. From the top view, rotate `Arm.R` slightly forward and `Arm.L` slightly back. Finish the pose with his left leg forward and right leg backward, both slightly bent. Gus is walking in place!



Our second pose.



Pose menu.

- Select all bones again and press I to store this pose at frame **11**, and select Rot.
- We now need a third pose at frame **21**, with the right leg up, because we are in the middle of the other half of the stride. This pose is the mirror of the one we defined at frame **1**. Therefore, return to frame **1** and, with all the bones selected, in the Pose menu of the 3D window header, select the Copy Pose entry, see (*Pose menu*). Or use CtrlC. You have now copied the current pose to the buffer.
- Go to frame **21** and paste the pose with the Paste X-Flipped Pose option in the Pose menu, see (*Pose menu*). Or use ⇧ ShiftCtrlV. This will paste the cut pose, exchanging the positions of bones with suffix "`.L`" with those of bones with suffix "`.R`", effectively flipping it!

Bone roll
If pasting an X-flipped pose makes Gus bend in the wrong direction you may have some trouble with the Bone Roll. Select all bones in Edit Mode and press CtrlN, 3 to sort out the bone roll. Then go back to frame 1 and 11 and adjust the poses. Re-copy the pose from frame 1 and again try pasting the x-flipped pose in frame 21.

The pose is there but it has not been stored yet! You must press I » Rot with all bones selected.

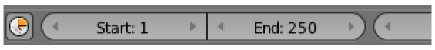
- Now apply the same procedure to copy the pose at frame **11** to frame **31**, also flipping it.
- To complete the cycle, we need to copy the pose at frame **1**, *without* flipping it, to frame **41**. Do so by copying it as usual, and by using the Paste Pose entry. Or use CtrlV. End the sequence by storing the pose with I » Rot.

Checking the animation
To preview your animation, set the current frame to **1** and press AltA in the 3D window.

Gus walks!

The single step in-place is the core of a walk, and once you have defined one there are techniques to make a character walk along a complex path. But, for the purpose of our Quick Start, this single step in-place is enough.

- In the Render context in the Properties window, set the start frame (Start) to **1** (it should already be at **1** by default) and set the end frame (End) to **40** (it is set to **250** by default, see *Settings for an animation in the Render context*). Because frame **41** is identical to frame **1**, we only need to render frames from **1** to **40** to produce the full cycle.



Settings for an animation in the Render context.

- Type *//render/* in the text field in the Output panel.
- Select AVI Raw as the file type in the Format panel. While this is generally not the best choice, mainly for file size issues, it is

fast and it will run on any machine, so it suits our needs. You could also select AVI Jpeg to produce a more compact file. However, it uses lossy JPEG compression and will produce a movie that some external players might not be able to play.

Finally, press the Animation button in the Render panel. Remember that *all* the layers that you want to use in the animation must be shown! In our case, these are layers 1 and 10.

Stopping a Rendering
If you make a mistake, like forgetting to turn layer 10 on, you can stop the rendering process with Esc.

Our scene is pretty simple, and Blender will probably render each of the forty images in a few seconds. Watch them as they appear.

Stills
Of course you can always render each of your animation frames as a still by selecting the frame you wish to render and pressing the RENDER button.

Once the rendering is complete you should have a file named `0001_0040.avi` in a `render` subdirectory of your current directory – the one containing your `.blend` file. The directory can be changed from the Output panel.

You can play this file directly within Blender by pressing Play Rendered Animation in the top menu (or by using CtrlF11). The animation will automatically cycle. To stop it press Esc. We have produced only a very basic walk cycle. There is much more in Blender, as you'll soon discover reading its whole manual!

File operations

The options to manage files are:

New	Clears the current scene and loads startup.blend
Open	Open a blend file
Open Recent	Displays a list of recently saved .blend files to open
Recover last session	This will load the quit.blend file Blender automatically saves just before exiting. So this option enables you to recover your last work session, e.g. if you closed Blender by accident
Recover Auto Save	This will open an automatically saved file to recover it.
Save	Save the current blend file.
Save As	Opens file browser to specify file name and location of save.
Save Copy	Saves a copy of the current file.
User Preferences	Opens the user preferences dialogue.
Save User Settings	Saves the current scene and preferences to startup.blend.
Load Factory Settings	Restore the default scene to the factory settings.
Link or Append	You don’t have to load a complete file, you can load in only selected parts from another file if you wish.
Import, (COLLADA)	Blender can use information stored in a variety of other format files which are created by other graphics programs.
Export	Normally you save your work in a .blend file, but you can export some or all of your work to a format that can be processed by other graphics programs.
External Data	Pack into .blend Pack all used external files into the .blend Unpack into Files Unpack all files packed into this .blend to external ones Make all paths Relative Make all paths to external files relative to current .blend Make all paths Absolute Make all paths to external files absolute Report Missing Files Report all missing external files Find Missing Fils Try to find missing external files

Opening Files

Mode: All modes

Hotkey: F1

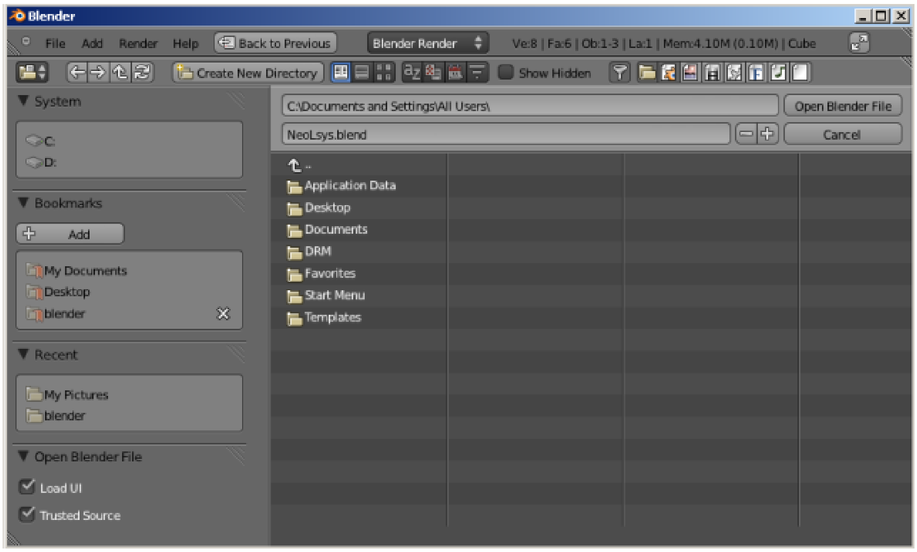
Menu: File » Open

Description

Blender uses the `.blend` file format to save nearly everything: objects, scenes, textures, and even all your user interface window settings.

Blender expects that you know what you are doing! When you load a file, you are **not** asked to save unsaved changes to the scene you were previously working on, completing the file load dialog is regarded as being enough confirmation that you didn't do this by accident.

Make sure that you save your files.



Using the File Browser and Folder Navigation

To load a Blender file from disk, press F1. The File Browser window, as shown above, will open.

The upper text box displays the current directory path, and the lower text box contains the selected filename. P (or the P button) moves you up to the parent directory.

The + and - buttons allow you to cycle through numbered files by increasing or decreasing the number at the end of the file name.

Click on a folder to go inside of it, or click on a file then press the Open Blender File to open it

Clicking Cancel will close the file browser window and return to the program.

Side Panel

The panel on the left displays different ways to find files and several options. To load a file, select it with LMB and then press ↵ Enter, or click the Open File button. A file can also be loaded by simply clicking MMB over its name.

System

The system menu contains a list of drives that are available to navigate through to find files. Click on one to jump to that drive.

Bookmarks

These are folders that you want to be able to access often without having to navigate to them in the file browser. To add a directory to the bookmark menu, navigate to that folder, then click the Add button. To remove a folder from the list, simply click the X icon nexto to it.

Recent

This is a list of recently accessed folders. You can control how many folders appear in this list by going to the File tab of the user Preferences, in the box labeled Recent Files.

Open Options

Inside each .blend file, Blender saves the user interface – the screen layouts. By default, this saved UI is loaded, overriding any user defaults or current screen layouts that you have. If you want to work on the blend file using your current defaults, start a fresh Blender, then open the file browser (F1). Turn off the Load UI button, and then open the file.

The Header Panel

The Header contains several tools for navigation files. The four arrow icons allow you to:

- **Move to previous folder**
- **Move to next folder**
- **Move up to parent directory**
- **Refresh current folder**

Create a new folder inside the current one by clicking the Create New Directory icon.

The other icons allow you to control what files are visible and how they are displayed. You can:

- **Display files as a short list**
- **Display files as a detailed list**
- **Display files as thumbnails**

You can sort files:

- **Alphabetically**
- **By file type**
- **By Date of last edit**
- **By file size**

Filtering controls which file types are shown. Click the Enable Filtering icon, and toggle which types are shown:

- **Folders**
- **Blend files**
- **Images**
- **Movie files**
- **Scripts**
- **Font files**
- **Music files**
- **Text files**

Other File Open Options

From the File menu, you can also open files with the following tools:

Open Recent
Lists recently used files. Click on one to load it in.
Recover Last Session
This will load the `quit.blend` file Blender automatically saves just before exiting. So this option enables you to recover your last work session, e.g. if you closed Blender by accident…
Recover Auto Save
This will open an automatically saved file to recover it.

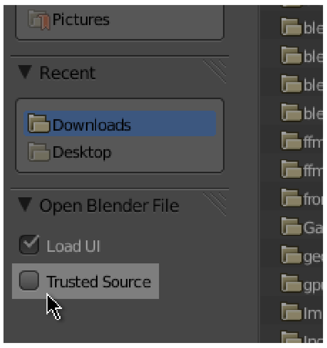
Security

Blender is aimed at production level use and relies heavily on Python, a powerful scripting language. Python can be used in Blender to create new tools, importers and exporters, and also to drive animation rigs. With Python scripting there are endless possibilities in what you can create with Blender.

Part of Python's power comes from having full access to your system, however this power can also be misused in the wrong hands. It's possible (but not terribly likely) for dishonest people to distribute .blend files containing scripts that may damage your system. These scripts can be attached as part of animation rigs, so that they will be run when such a .blend file is opened.

Always be very careful when downloading .blend files and tools from un-trustworthy sources!

Protection



To protect against malicious .blend files, it's possible to prevent any embedded scripts from running when you open a .blend file. This will mean that custom tools or rigs using Python features will not work, but this won't be a problem for .blend files that don't use these (such as material libraries), and will at least give you a chance to better evaluate what risks might be inside.

By default, Blender will trust all files and run scripts automatically. If you don't trust the file, and want protection, you can disable 'Trusted source' in the File->Open dialog in the properties section on the bottom left. Un-trusted files will disable embedded Python scripts after opening the file.

Saving Files

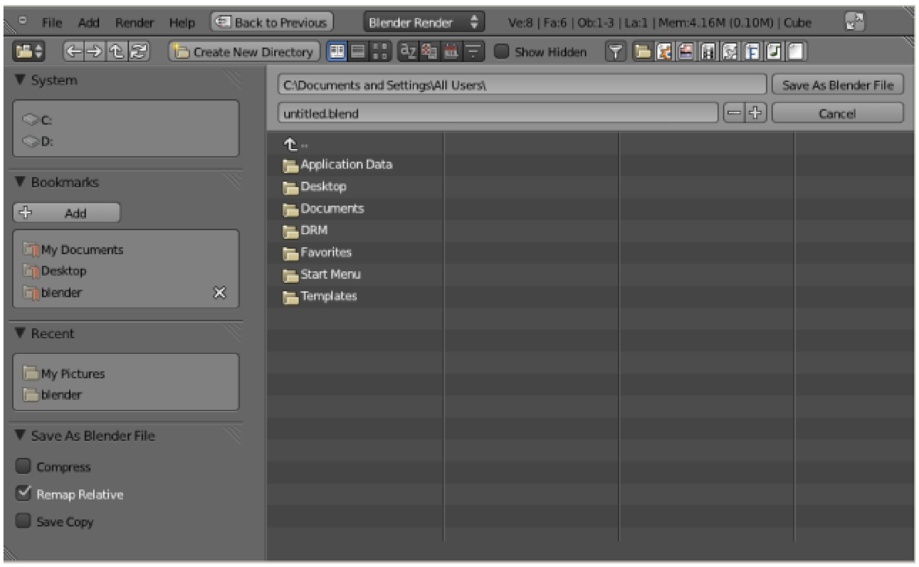
Mode: All modes

Hotkey: F2

Menu: File » Save

Description

Saving files is like loading files. When you press F2, File Browser window will open. The window appears the same as when opening files, except for a few options in the side panel. For descriptions and usage of the file browser functions, see the page on Opening files.



Saving

Click the lower edit box to enter a filename. If it doesn't end with "`.blend`", the extension is automatically appended. Then press ↵ Enter or click the Save File button to save the file.

When saving via menu or pressing F2, Ctrl⇧ ShiftS or CtrlAltS, if a file with the same given name already exists, the edit box will turn red as a warning, if you press Enter or the Save button, it will automatically save over the existing file with the same name. So, think twice before you continue saving.

By default, when pressing CtrlS or CtrlW, it will pop-up a message for you to confirm your saving, which aims to avoid the "accidental" saving operation, which may not be what you want. (e.g. say, you want to press Ctrl A when operating, but the Ctrl S is pressed, then you can just move your mouse cursor out of the pop-up, the accidental saving operation will be canceled.)

Depending on the number of *Save Versions* you have set, all existing files with the same name will be rotated to a `.blend`*n* file extension, where *n* is `1`, `2`, `3`, etc. So, if you were working on `MyWork.blend`, and saved it, the existing `MyWork.blend` is renamed to `MyWork.blend1`, and a new `MyWork.blend` is saved. This way, you have hot backups of old saved versions that you can open if you need to massively undo changes.

Save Options

The save options appear at the bottom of the side panel.

Compress File

Enable this option to squash large files, this removes dead space.

Remap Relative

This option remaps relative paths when saving a file in a new location

Save Copy

This option saves a copy of the actual working state, but does not make the saved file active.

Tip for Save Increments

The save dialog contains a little feature to help you to create multiple versions of your work: pressing + NumPad or - NumPad increments or decrements a number at the end of the file name. To simply save over the currently loaded file and skip the save dialog, press CtrlW instead of F2 and just confirm at the prompt.

Importing Files

Blender allows you to import other file formats

Collada (.dae)

See Here

Motion Capture (.bvh)

Target
: Import an Object or Armature.

Scale
: Scale the BVH by this value

Start Frame
: Starting frame for the animation

Loop
: Loop the animation playback

Rotation
: Rotation conversion. Uses either Quaternion or Euler

Forward
: Specify the forward axis

Up
: Specify the up axis

Scalable Vector Graphics

Imports a vector graphic as a curve object

Stanford (.ply)

See Here

Stl (.stl)

See Here

3d studio (.3ds)

See Here

Size Constraint
: Scale the model by 10 until it reaches the size constraint (0 to disable)

Image Search
: Search sub directories for any associated images.

Apply Transform
: Workaround for object transformations importing incorrectly

Forward
: Specify the forward axis

Up
: Specify the up axis

Autodesk FBX (.fbx)

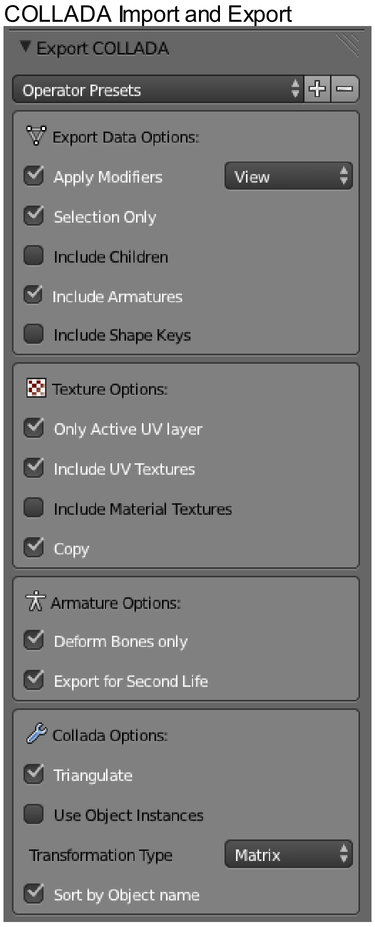
See Here

Wavefront (.obj)

See Here

X3D Extensible 3D (.x3d)

COLLADA Import and Export


Export COLLADA
Operator Presets
Export Data Options:
- [✓] Apply Modifiers — View
- [✓] Selection Only
- [ ] Include Children
- [✓] Include Armatures
- [ ] Include Shape Keys

Texture Options:
- [✓] Only Active UV layer
- [✓] Include UV Textures
- [ ] Include Material Textures
- [✓] Copy

Armature Options:
- [✓] Deform Bones only
- [✓] Export for Second Life

Collada Options:
- [✓] Triangulate
- [ ] Use Object Instances
- Transformation Type — Matrix
- [✓] Sort by Object name

The Collada module has been implemented as a flexible tool for exporting and importing .dae files. We have taken care to provide a set of parameters which should make it possible to export/import Collada files from/to a variety of tools. But please be aware that we are not yet finished with the Collada module. So it may well be possible that your particular usage scenario is not yet supported.

The Collada Exporter

Operator Presets

We have added 2 Operator Presets (see top of option panel) for Second Life users:

- **Second Life Static**
  is good for exporting static meshes.
- **Second Life Rigged**
  is good for exporting the SL default character.

Special Notes for Second Life users:

- Please use the Operator presets. All other export settings will not work for Second Life.
- The character orientation needs to be such that the character looks towards positive X.
- Scale&Rotation must be applied before export!

Export Data Options

Apply Modifiers

All active Modifiers will be applied in a non destructive mode. That is, the modifiers will be applied to copies of the meshes. Thus you no longer need to apply your modifiers before exporting. That is now done automatically in the background.

Preview/Render mode
Some Modifiers provide a Preview mode and a Render mode with different mesh settings. We now support both modes when applying the modifiers.

Selection Only

When selection only is enabled, then only the selected objects will be exported. Otherwise the entire scene is exported with all visible and all invisible objects.

Include Children

When this option is enabled then all children of the selected objects will also be exported regardless of their selection state.

Example
You now can select only(!) an armature, then in the exporter enable “include children” then all rigged meshes attached to the armature will also be exported.

Include Armatures

When this option is enabled, then all armatures related to the selected objects will also be exported regardless of their selection state.

Example
You now can just select your objects, then in the exporter enable “include armatures” then the armature data is also exported.

Include Shape keys

Shape keys
This option also includes the application of Shape keys! So now you can export your meshes with the current shape key configuration baked in.

Texture Options

Only Active UV layer

When your mesh contains multiple UV layers, then Blender exports all layers by default. This option allows you to only export the active UV layers.

Include Textures

Blender supports 2 ways to texturise your objects.

1. By using directly assigned surface textures
2. By using material based image textures

While the material based image textures offer much more flexibility, using surface textures can be done very quickly without need to first render textures. Until now blender did only export material based image textures. The new option allows to directly export render results.

Texture export needs materials
For using surface textures, you will still have to create a material for each texture face. Then all you need to do is assign your images to the correct faces of your mesh. And finally when your object looks as you expect, just export it with “Include UV Textures”. See also the “Copy” option below.

Copy

When you export images either material based image textures or surface textures, then we create absolute file references in the export file.

But if the “Copy” option is enabled, we will create copies of the images instead and place the copies besides the export file. In that case the file references are made relative.

Armature Options

Deform Bones Only

When this option is enabled, then the exporter strips all non deformiung bones from the exported armatures. This option is useful when your armatures contain control bones which are not actually part of the charater skeleton. For example you can now export the Avastar rig with this option enabled. The resulting exported rig is compatible to Second life. But please note the restrictions further down.

Export for Second Life

This option is very special. In fact some issues with bone orientation are calculated differently when this option is enabled. This is only relevant for rigged meshes. I hope that this option will eventually be replaced by something more meaningful (and still compatible to Second Life) Hint: This option is only important when you want to export rigged meshes. For static meshes it just does nothing at all.

Collada Options

Triangulate

The Mesh con be triangulated on the Fly. The triangulation is based on the same function which is used in the User interface for triangulating the current selection of faces. For full control over the triangulation you can do this manually before exporting. However this option allows to do the triangulation only for the exported data. The mesh itself is not affected.

Use Object Instances

In Blender you can reuse the same mesh for multiple Objects. This is named “object instanciation”. When you enable this option, then Blender will propagate object instantiation to the Collada file.

Transformation Type

Collada supports 2 types of Transformation matrix specifications. Either as <Matrix> or as a set of transformation decompositions (for Translate, Rotate and Scale). Note that the exporter will not strictly follow this option setting, but will rather take it as a hint to use the option if ever possible. This is so because some of the exported data types hae specific rules about how the transformation matrix has to be exported. This is ongoing development and we may provide a less ambiguous method in the future.

Sort by Object Name

The export order of data is bound to internal object order and it can not be influenced in a reliable way. this option ensures that the Geometry nodes and the Object nodes are both exported in alphabetical order.

The Collada Importer

The Collada Importer is mostly driven by the imported Data. We only have added one option for controlling the Import units:

- Import Units

If not enabled the imported data will be rescaled according to the currently used unit system. We assume that the Blender unit is 1 meter. if this option is enabled, then Blender will adjust itself to the unit system as provided by the Collada file.

Technical details

Mesh

Import

Supported geometry types are

- tris (not tested)
- polylist
- polygons
- ngons
- trifans (not tested)
- lines

Export

Mesh data is exported as <polylist>, <lines> and <vertices>.

Light

Import

Blender does a best effort on importing lights from a .dae. If a Blender profile is detected for lights, all values from these will be used instead. This ensures 100% reimport from a Blender exported .dae. <extra> support has been added in Blender 2.57.

Export

A Blender profile for lights has been added through the <extra> tag. The entire Lamp struct from Blender will be exported through this profile, with the exception of Light curve falloff .

Material & Effect

Export

Since Blender 2.57 some changes to export of effects have been made. Most notably <lambert> is exported if and only if specularity is 0.

Animation

Export&Import

- Support for Object(Mesh, Camera, Light) transform Animations. Only euler rotations, which is the default option for Objects, can be exported for now. For armature bone animations euler and quaternion rotation types are supported.
- Import and export of animations for the following parameters are supported:-
  - Light
  - Camera
  - Material Effects
- Non Skin controlling armature bone animation.
- Animations of Armatures with skin deforming bones.
- Animations of Armatures in Object mode.
- Fully rigified Armature animations. For export of rigified Armature animations
  - Select Bake Action. (press space in 3d view and Type Bake Action)

- If you have only the deform bones selected check "only selected". This will give smaller dae. Otherwise uncheck "Only Selected".
- Check "Clear Constraints".
- Bake Action.
- Select the mesh and the deform bones. Then export to COLLADA while checking only selected option. (Selecting only the Mesh and bones is not strictly necessary. Selecting and export only selected will give smaller dae.)
- Demonstration

Nodes

On import parent transformations for <instance_node>s is properly propagated to child node instances. Blender materials are exported with the following mapping:

- phong
- blinn
- lambert

For bone nodes which are leaf nodes in the armature tree, or if a bone has more than one children a blender profile for tip with an <extra> tag, is added for those joint nodes. To correctly derive the bone->tail location on re-import.

Important things to remember

- object and datablock names are constrained to 21 characters (bytes).
- uv layer names are constrained to 32 characters (bytes).
- only armature animation on mesh, single skin controller
- no support for modifiers yet

When importing a .dae that has <instance_node>s on exporting this information is essentially lost and these nodes will be <node>s.

Exporting Files

Its not always that you will have to work on a project using one software. Basically, you might start off in Blender, but you will want to switch to another software and continue from there.

That is possible as Blender allows *Exporting* to other formats. Blender 2.65 currently supports exporting to 8 formats, including COLLADA.

These are important so that you can share you files with others who are not Blender users, such as those using Autodesk® 3DS Max®.

Collada (.dae)

See Here

Motion Capture (.bvh)

only available for import

Stanford (.ply)

See Here

Apply Modifiers
: Bakes down the modifier stack by applying them.

UV's
: Exports active uv layer's coordinates of the geometry.

Normal
: Exports the geometry normal data for smooth and hard edges.

Vertex Colors
: Exports the active vertex color layer

Stl (.stl)

See Here

Ascii
: Saves the file in ASCII file format.

Autodesk® 3DS Max® (.3ds)

See Here

Selection Only
: Export selected objects only

Forward
: Specify the forward axis

Up
: Specify the up axis

Autodesk® FBX (.fbx)

See Here

Wavefront (.obj)

See Here

X3D Extensible 3D (.x3d)

Supported Formats

Image Formats

This is the list of image file formats supported internally by Blender:

High Dynamic Range Graphics

Blender's image input/output system transparently support regular 32 bits graphics (4 x 8 bits) or floating point images storing 128 bits per pixel (4 x 32 bits).

On reading High Dynamic Range Images (HDRI), even when they're for example 3 x 10 bits, the pixels are always converted internally to RGBA float values. Optionally, like while displaying the image in the UV Image editor, this then gets converted to regular 32 bits for faster display.

When an image has float colors, all imaging functions in Blender default to use that. This includes the Video Sequence Editor, texture mapping, background images, and the Compositor.

For hints how to manipulate or view HDR images, please check the Curves UI page.

- OpenEXR (Multilayer)
- DPX, Cineon and Radiance HDR

Others Formats

Here's a list of other supported image formats:

- BMP
- DDS
- SGI IRIS (old!)
- PNG
- JPEG
- JPEG 2000
- Targa
- Targa Raw (uncompressed Targa)
- TIFF

Movie Formats

- AVI (Windows)
- AVI JPEG
- AVI Raw
- Frame Server
- H.264
- MPEG
- Ogg Theora
- QuickTime
- Xvid

Color Modes

- BW, Images get saved in 8 bits grayscale (only PNG, JPEG, TGA, TIF)
- RGB, Images are saved with RGB (color)
- RGBA, Images are saved with RGB and Alpha data (if supported)

Color Depths

- 8 bit color channels
- 12 bit color channels
- 16 bit color channels
- 32 bit color channels

Reference

- Formats

OpenEXR

ILM's OpenEXR has become a software industry standard for HDR image files, especially because of its flexible and expandable structure.

OpenEXR files can store values in the entire floating point space, positive as well as negative numbers.

Apart from reading all OpenEXR file types, with RGBA and optional Z saved, Blender supports OpenEXR in two ways:

- Render Output
- Multi-layer, Multi-pass, tile-based files

Render Output

Available options for OpenEXR render output are:

Half
Saves images in a custom 16 bits per channel floating point format. This reduces the actual "bit depth" to 10 bits, with a 5 bits power value and 1 bit sign.

Zbuf
Save the depth information. In Blender this now is written in floats too, denoting the exact distance from the camera in "Blender unit" values.

Preview
On rendering animations (or single frames via command line), Blender saves the same image also as a JPEG, for quick preview or download.

Compression (this button is below the Image menu button, default set to "None")

- PIZ, lossless wavelet compression. Compresses images with grain well.
- ZIP, standard lossless compression using zlib
- RLE, runlength encoded, lossless, works well when scanlines have same values.
- PXR24. lossy algorithm from Pixar, converting 32 bits floats to 24 bits floats.

Multi-layer, Multi-pass, tile-based files

An OpenEXR file can hold unlimited layers and passes, stored hierarchically. This feature now is in use for the "Save Buffers" render option. This option doesn't allocate the entire final Render Result before render (which can have many layers and passes), but saves for each tile the intermediate result to a single OpenEXR file in the default Blender 'temp' directory.

When rendering is finished, after all render data has been freed, this then is read back entirely in memory.

DPX and Cineon

Cineon is Kodak's standard for film scanning, 10 bits/channel and logarithmic. DPX has been derived from Cineon as the ANSI/SMPTE industry standard. DPX supports 16 bits color/channel, linear as well as logarithmic. DPX is currently a widely adopted standard used in the film hardware/software industry.

DPX as well as Cineon only stores and converts the "visible" color range of values between 0.0 and 1.0 (as result of rendering or composite). No alpha is written.

The code has been gratefully copied from CinePaint. According to CinePaint's main developer Robin Rowe the DPX code defaults to logarithmic colorspace. We cannot find yet how to disable this, but it seems to read/write fine without visible loss.

Blender DPX files (entire Elephants Dream movie) have been succesfully imported in a Quantel IQ HD/2K finishing/grading set without problems, so for now we assume it's compliant for professional usage.

Radiance HDR

Radiance is a suite of tools for lighting simulation originally written by Greg Ward. Since Radiance had the first (and for a long time the only) HDR image format, this format is supported by many other software packages.

Radiance (.hdr) files store colors still in 8 bits per component, but with an additional (shared) 8 bits exponent value, making it 32 bits per pixel. Only RGB can be stored in these files.

Overview

Each .blend file contains a database. This database contains all scenes, objects, meshes, textures, etc. that are in the file. A file can contain multiple scenes and each scene can contain multiple objects. Objects can contain multiple materials which can contain many textures. It is also possible to create links between different objects.

Mode: All Modes, Any Window

Hotkey: ⇧ ShiftF4 - Data Select Browser

To access the database, press ⇧ ShiftF4 and the window will change to a Data Select Browser window, which lists the Objects in your .blend file. To go up a level, click the breadcrumbs (. .) and then you will see the overall structure of a file: Action, Armature, Brush, Camera, Curve, Group, and so on (including Objects).

LMB selecting any datablock type, Mesh, for example, will give you a listing of the meshes used in the file, along with how many users there are for each one. For example, if you had a car mesh, and used that car mesh for six cars in a parking lot scene, the Mesh listing would show the Car and then the number 6.

Mode: Data Select Browser

Hotkey: F - Fake User

RMB selecting certain kinds of datablocks (Materials, Images, Textures…) and pressing F will assign a “fake user” to those datablocks. With a fake user in place, Blender will keep those datablocks in the file, even if they have no “real user”. Datablocks without a user, real or fake, are not saved in the .blend file. Pressing F again toggles the fake user assignment. Performing this action is the same as clicking the F button next to material/image/… names.

Outliner and OOPS Schematic

You can easily inspect the contents of your file by using the Outliner window. This window displays the Blender data system (fully documented here). This window offers two views of the database. The Outliner view allows you to do simple operations on the objects. These operations include selecting, renaming, deleting and linking. The OOPS Schematic (Object-Oriented Programming System) view allows you to easily see how datablocks are linked. You can filter the view by using buttons found in the header.

Users (Sharing)

Many datablocks can be shared among other datablocks - re-use is encouraged. For example, suppose you have a material for one object, named “`Glossy`”. You can select a second object, for example, one that does not have a material yet. Instead of clicking ADD NEW for the material, click the little up-down arrow next to the ADD NEW, which brings up a list of existing materials. Select “`Glossy`”. Now, these two objects share the same material. You will notice a “2” next to the name of the material, indicating that there are two users (the two objects) for this material. Other common examples include:

- Sharing textures among materials.
- Sharing meshes between objects (“clones”).
- Sharing Ipo curves between objects, for example to make all the lights dim together.

Fake User

Remember that Blender does not save datablocks that are not linked to anything in the *current* file. If you're building a ".blend" file to serve as a library of things that you intend to link-to from *other* files, you'll need to make sure that they don't accidentally get deleted from the current (the library) file. Do this by giving the datablocks a "fake user," by hitting the F button next to the name of the datablock. This prevents the user count from ever becoming zero: therefore, the datablock will not be deleted. (Blender does not keep track of how many other files link to this one.)

Copying and Linking Objects Between Scenes

Sometimes you may want to link or copy objects between scenes. This is possible by first selecting objects you want to link or copy and then using the Make Links and Make Single User items found in Object menu in the 3D viewport header. Use Make Links to make links between scenes. To make a plain copy, you first make a link and then use Make Single User to make a stand-alone copy of the object in your current scene. Further information on working with scenes can be found here.

Appending or Linking Across Files

The content of one .blend file is easily accessed and put into your current file by using the File → Append function (accessed at any time by ⇧ ShiftF1). To find out more about how to copy or link objects across .blend files, click here.

Proxy Objects

Proxy objects allow you to make (parts of) linked data local. For example, this allows an animator to make a local “copy” of the handler bones of a character, without having the actual rig duplicated. This is especially useful for character animation setups, where you want the entire character to be loaded from an external library, but still permit the animator to work with poses and actions. Another example: you can have a modeler working on the shape (mesh) of a car and another painter working on the materials for that car. The painter cannot alter the shape of the car, but can start working with color schemes for the car. Updates made to the shape of the car are applied automatically to the painter’s proxy.

Pack and Unpack Data

Blender has the ability to encapsulate (incorporate) various kinds of data within the .blend file that is normally saved outside of the .blend file. For example, an image texture that is an external `.jpg` file can be put “inside” the .blend file via File → External Data → Pack into .blend file. When the .blend file is saved, a copy of that `.jpg` file is put inside the .blend file. The .blend file can then be copied or emailed anywhere, and the image texture moves with it.

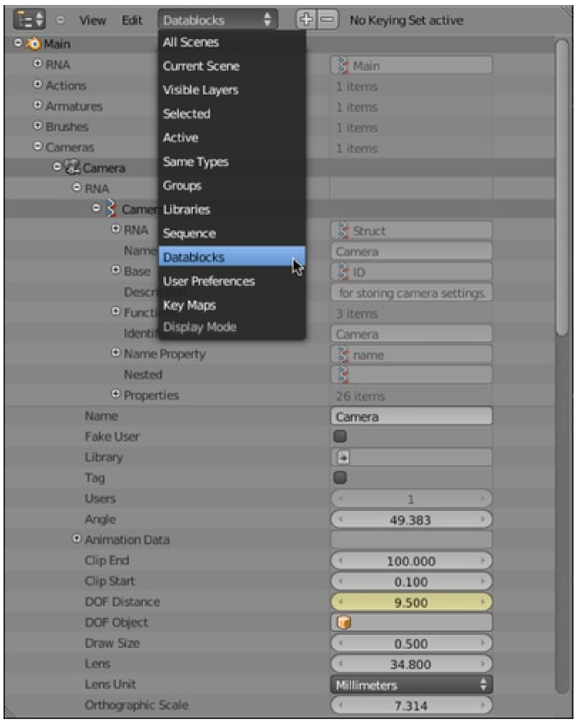
You know that an image texture is packed because you will see a little “Christmas present gift box” displayed in the header.

Unpack Data

When you have received a packed file, you can File → External Data → Unpack into Files.... You will be presented with the option to create the original directory structure or put the file in the `//` (directory where the .blend file is). Use “original locations” if you will be modifying the textures and re-packing and exchanging .blend files, so that when you send it back and the originator unpacks, his copies of the textures will be updated.

Datablocks

The base unit for any blender project is the datablock. Examples of datablocks include: meshes, objects, materials and textures. Be it a simple sphere floating over a plane, or a full featured film, a project is defined by the datablocks it contains, the properties set for those datablocks, and the way the datablocks link to each other. Datablocks can reside within as many .blend files as needed for good project organization.



RNA: Main
Actions: 1 items
Armatures: 1 items
Brushes: 1 items
Cameras: 1 items
RNA: Struct
Name: Camera
Base: ID
Description: for storing camera settings.
Functions: 3 items
Identifier: Camera
Name Property: name
Nested
Properties: 26 items
Name: Camera
Fake User
Library
Tag
Users: 1
Angle: 49.383
Animation Data
Clip End: 100.000
Clip Start: 0.100
DOF Distance: 9.500
DOF Object
Draw Size: 0.500
Lens: 34.800
Lens Unit: Millimeters
Orthographic Scale: 7.314

Datablocks view

Types of Datablocks:

- RNA
- Filename
- File Has Unsaved Changes
- File is Saved
- Cameras
- Scenes
- Objects
- Materials
- Node Groups
- Meshes
- Lamps
- Libraries
- Screens
- Window Managers
- Images
- Lattices
- Curves
- Metaballs
- Vector Fonts
- Textures
- Brushes
- Worlds
- Groups
- Shape Keys
- Scripts
- Texts
- Speakers
- Sounds
- Armatures
- Actions
- Particles
- Grease Pencil
- Movie Clips
- Masks

Copying and Linking Datablocks

It is possible to copy or link any type of datablock.

To copy a Scene datablock, use Scene list found in the header of Info window. The list is to the right of the menus and window workspace list. Select ADD NEW to make a copy of the current scene. Select Full Copy from the list that opened to make a copy. The current scene will be **fully** copied to the new scene.

Instead of copying **everything**, you can link datablocks by selecting Link Objects, to use the same Object datablocks linked into the two scenes, or Link ObData, to create separated objects sharing the same ObData datablocks (meshes, curves, materials, etc.). Note that if you select Link Objects, in fact you copy nearly nothing, as Object datablocks are parent of all ObData datablocks: nearly all modifications (object location, mesh geometry, …) in a scene will be reflected in the other ones linked this way as well. As for Link ObData, it creates a “real” copy of the objects, but not of their child datalocks: the locations, scales and rotations of the objects will be specific to a scene, but neither their forms nor their materials, textures, …, will be (as defined by ObData datablocks).

Copying and Linking Object Datablocks

Full copy
: ⇧ ShiftD is used to make normal copy of the selected objects.
: The object and some of it’s child datablocks will really be duplicated, the other children are just linked; you can define the attributes to be duplicated in User Preferences → Edit Methods, button group Duplicate with object:.

Linked copy
: AltD makes a linked copy.
: All datablocks but the object one are linked.

Copying and Linking other Datablocks

When an ObData datablock is used (linked) by more than one object, a small button with its number of linked objects (users) shows up next to its name, in its settings window (Editing context for meshes, curves, cameras, …, Shading context, Material sub-context for materials, etc.). If you click on it, you create a copy of this datablock for the current object.

Unlinked Datablocks

A datablock can become unlinked. For example a material datablock will be unlinked if the object it was linked to is deleted. If a datablock is unlinked, by default it will be deleted from the .blend file when Blender is closed. To keep an unlinked datablock in a .blend file, click the "F" button to the right of the object's name in the Objects and Links panel.

Scene Management Structure

Scene management and library appending/linking is based on Blender's Library and Data System, so it is a good idea to read that manual page first if you're not familiar with the basics of that system.

Blender can be used to create something as simple as a single scene or image, or scaled up to an entire movie. A movie is usually comprised of three acts:

1. Introduction-Conflict.
2. Rising Tension.
3. Climax-Resolution.

Each act contains a few scenes, settings where the action happens. Each scene is shot on a set, stage or location. Each is set with props and backdrops. The scene is a set of action sequences where the actors act (hopefully convincingly). Each sequence, or shot, usually lasts a few seconds.

Sequence shot
Sometimes, a single shot lasts several minutes: its a "sequence shot", which might even be a complete scene on its own. Technique hard to master if you don't want your audience to fall asleep!

A single Blender file is organized and set up to be able to contain an entire movie. Each .blend file can contain multiple scenes. A scene is a bunch of objects, organized into layers. As you progress through the creative process, you use a set of window screen layouts specifically designed to help you work efficiently through the creative process: model the objects and create the props, clothe the actors and dress the set (assign materials), define the action (animation), render the video, and produce the movie. You can tailor these screen layouts, and create custom layouts, to match your working preferences.

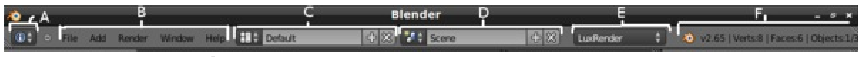
Planning Your Timeline

Shots within a scene are accomplished by moving a camera and/or actors through the scene for a few seconds. Time in Blender is measured in frames by default (even though you can change to seconds too if you wish), and typical video has 25 or 30 frames per second (fps), and film is 24 fps. For a five-second shot then, you allocate up to 150 frames for that shot (30 fps × 5 seconds). Giving yourself some wiggle-room, shot 2 would start at frame 250 and go from there. A one-minute movie set in a single scene for North America video broadcast (NTSC standard) would have a timeline that goes up to 1800 final frames, and may be laid out over the course of 2500 frames. This timeline allows for cutting out 700 frames, picking the best 1800 frames (30 fps × 60 seconds = 1800 frames) less transition time.

Multiple Cameras
You can have multiple cameras in a scene, used for different shots, and select which one is active when rendering the shot. Press [Ctrl + 0] to switch to the camera you wish to use at a point in time.

Current Screen Layout and Scene

Scenes are a way to organize your work. Scenes can share objects, but can, for example, differ from one another in their rendered resolution or their camera view. The current window layout and scene are shown in the Info window header, usually displayed at the top of your screen:


A
B
C
Blender
D
E
F
File Add Render Window Help
Default
Scene
LuxRender
v2.65 | Verts:8 | Faces:6 | Objects:1/3

Info window header. **A**) Window type icon,
B) Menu, **C**) Screen Layouts, **D**) Scenes, **E**) Renderer Options
F) Version of Blender currently running (click the Blender icon to the left to show splash screen).

Loading the UI with “File” → “Open”

Inside each .blend file, Blender saves the user interface layout - the arrangement of screen layouts when the file is saved. By default, this saved UI is loaded, over-riding any user defaults or the current screen layout. However, you can work on a blend file using your current UI settings by ignoring the UI settings saved in the file. This is done by restarting Blender or resetting it with (File → New, or CtrlX), and opening the file browser with (File → Open..., or F1). Turn off the Load UI button in the file browser header, and then open the file. This way, Blender will not disturb your current screen layout when it loads the new file.

Working with Scenes

Select a scene to work on by clicking the up-down arrow next to the scene name. Scenes and the objects they contain are generally specific to the project you are working on. However, they too can be saved in their current state to be re-used by pressing CtrlU. They will then appear the next time Blender starts or when the user selects File → New (CtrlX).

Blender comes with one default scene, which contains a camera, a lamp, and a box.

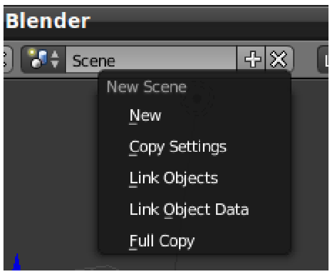
Adding a Scene

You can make a full copy of the current scene, start over with a blank slate, or create a scene that has links back to the current scene; objects will show up in the new scene, but will actually exist in the old one. Use this linking feature when, for example, the original scene contains the set, and the new scene is to contain the actors or props.

Starting Over
If you start with a new scene, be sure to add a camera and lights first!

Scenes are listed alphabetically in the drop-down list. If you want them to appear in a different order, start them with a numerical ordinal, like “`1-`”. The internal reference for a scene is the three-letter abbreviation “SCE”.

To add a scene, click on the scene list button, and select Add New. While you are adding a new scene, you have these options:



Add scene popup menu.

Empty
: Create a completely empty scene.

Link Objects
: All objects are linked to the new scene. The layer and selection flags of the objects can be configured differently for each scene.

Link ObData
: Duplicates objects only. ObData linked to the objects, e.g. mesh and curve, are not duplicated.

Full Copy
: Everything is duplicated.

Usually, for your first scene, you make a full copy of the default. Alternatively, you can just start with the default, and start editing the cube that is usually hanging around waiting for you to do creative things.

Naming a Scene

By ⇧ Shift LMB-clicking on the scene name (usually “`Scene.001`”), you can change the name of the scene. For example, “`BoyMeetsGirl`” is usually the first of three acts.

You then proceed to model the props and objects in the scene using the 2-Model window layout.

Linking to a Scene

You can, at any moment, link any object from one scene to another. Just open the scene where these objects are, use CtrlL → To Scene..., and choose the scene where you want your objects to appear. Those will be linked to the original objects; to make them single user (independent, unlinked…) in a given scene go to that scene, select them and use U. You will be presented with a few options that allow you to free up the datablocks (Object, Material, Texture…) that you want.

Removing a scene from the file

You can delete the current scene by clicking the X next to the name.

Outliner window

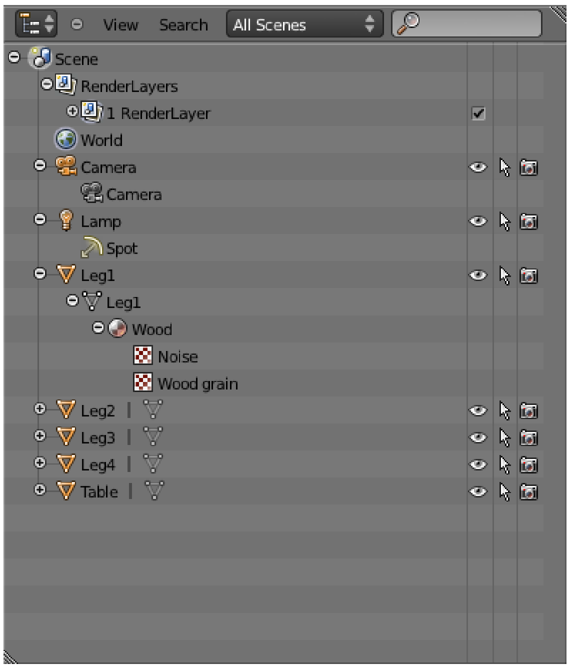
Description

The Outliner window is used for easily navigating a complex scene. The Outliner gives you a 2D representation of your complicated 3D world. Use it to find things in your scene.

For example, suppose you sneeze while moving an object; your mouse flies off your desk (gesundheit!) and the object is hurled somewhere off screen into space. Simply use the outliner to find it; select it, and move back to your 3D window to snap it back to your cursor (⇧ ShiftS → Selection → Cursor).

Another more practical example is to evaluate the impact of a change on related datablocks. Suppose you are looking at your `TableTop` object, and it doesn't look right, the `Wood` material doesn't look right; you want it to look more like mahogany. Since the same material can be used by many meshes, you're not sure how many things will change color when you change the material. Using the Outliner, you could find that material and trace the links that it has to every mesh in your scene.

Outliner view

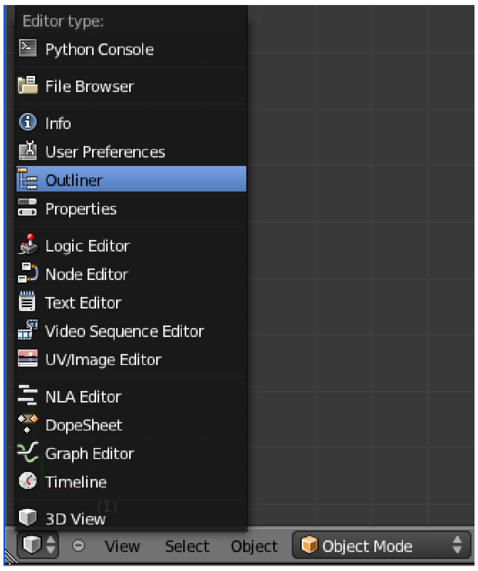


The Outliner window.

The Outliner is a kind of list that organizes things related to each other. In the outliner, you can:

- View the data in the scene.
- Select and deselect objects in the scene.
- Hide or show an object in the scene.
- Enable or disable selection (to make an object “unselectable” in the 3D Views).
- Enable or disable the rendering of an object.
- Delete objects from the scene.
- Unlink data (equivalent to pressing the X button next to the name of a datablock).
- Easily select which render layer to render.
- Easily select which render pass to render (for example, you can choose to render just the Specular pass).

Selecting the outliner window type



Change a window type to the Outliner.

Choose a window and LMB on its current Window type button (left-most icon in its header), and LMB on Outliner.

Using the Outliner

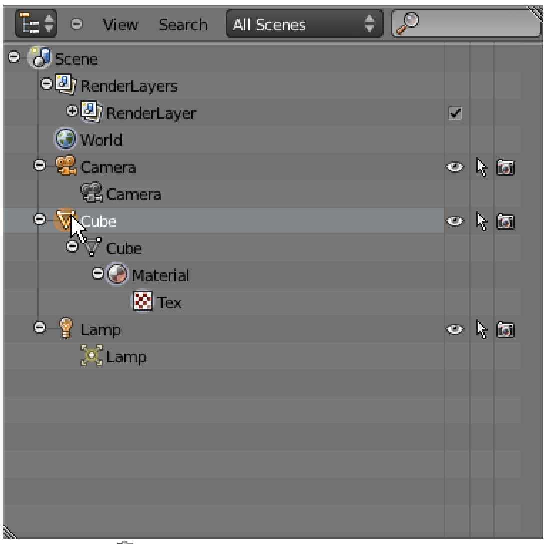
Each row in the Outliner shows a datablock. You can click the plus-sign to the left of a name to expand the current datablock and see what other datablocks it contains. If the row is already expanded, the icon to the left will be a minus-sign. Clicking the minus-sign will collapse subordinate objects beneath the row.

You can select datablocks in the Outliner, but this won't necessarily select the datablock in the scene. To select the datablock in the scene, you have to activate it.

Selecting and activating

Single selection doesn't require any pre-selection: just work directly with LMB (and/or RMB - contextual menu, see below) *inside* the name/icon area.

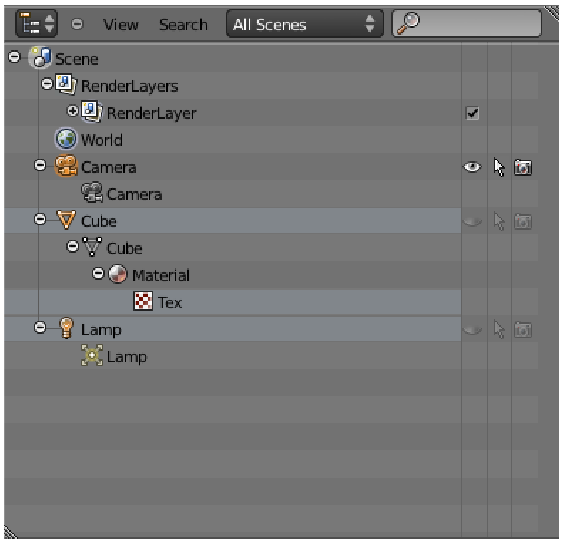
When you select an object in the list this way, it is selected and becomes the active object in all other 3D Views. Use this feature to find objects in your 3D View, select them in the Outliner, then zoom to them with . NumPad or if you don't have a numpad, snap and center your cursor on them via ⇧ ShiftS → Cursor → Selection, and then ⇧ ShiftC.



Click LMB on the mesh data of the cube to activate Edit mode.

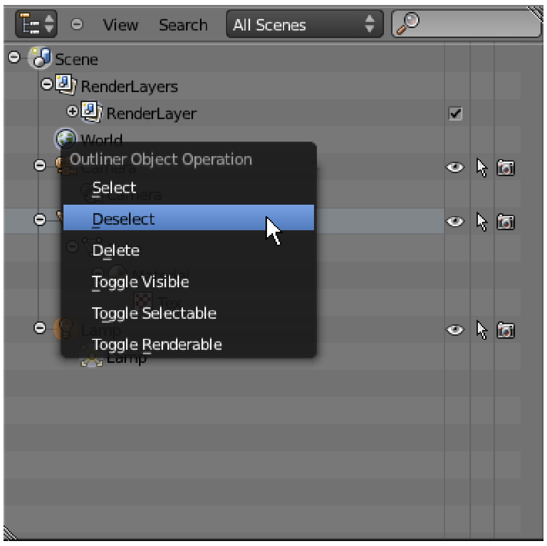
Activating a datablock

Activate the datablock with LMB on the *icon* of the datablock. Activating the datablock will automatically switch to the relevant mode. For example, activating the mesh data of the cube will select the cube and enter Edit mode while activating the object data of the cube will select the cube and enter Object mode (see right).



Toggling pre-selection of a datablock.

Toggle pre-selection of a group of datablocks
: Useful when you want to select/deselect a whole bunch of datablocks. For this you must prepare the selection using LMB *outside* the name/icon area. Those pre-selected have their line in a lighter color.
You then can (de)select them with a RMB *on* the name/icon area, which brings on a context menu (see bellow).



Context menu for the `Cube` object.

Context menu
: Show the context menu for a datablock with RMB on the icon or name. Depending on the type of the pre-selected datablock(s), you will have all or part of the following options:

- Select.
- Deselect.
- Delete.
- Unlink – To unlink a datablock from its “owner” (e.g., a material from its mesh).
- Make Local – To create a “local” duplicate of this datablock.

Note: some datablock types will not have a context menu at all!

Deleting a datablock
: Use X to delete the selected datablock(s).

Expanding one level
: Use + NumPad to expand one level down in the tree-list.

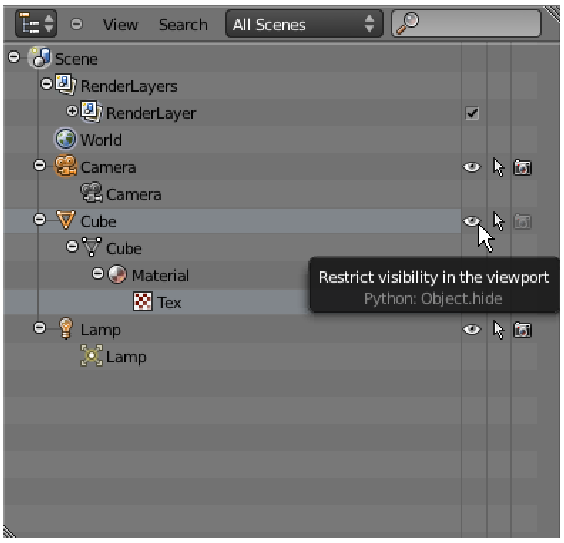
Collapsing one level
: Use - NumPad to collapse one level up in the tree-list.

Expanding/collapsing everything
: Use A to expand/collapse all levels of the tree-list.

Toggling object-level restrictions

The three following options, in the right side of the Outliner window, are only available for objects:

Visibility

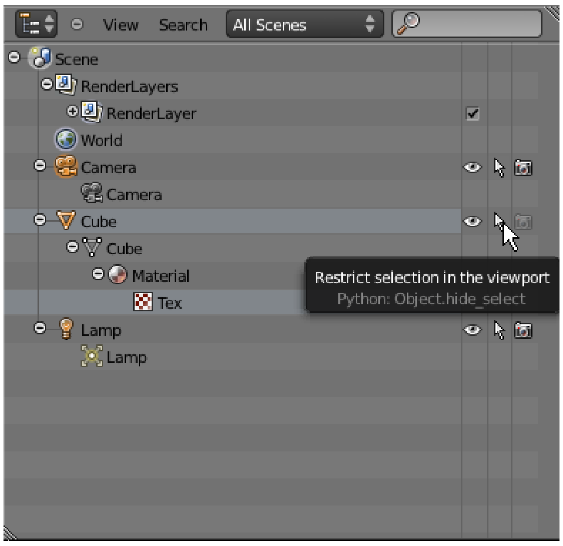


Restrict visibility in the viewport
Python: Object.hide

Restrict visibility

Toggle visibility by clicking the “eye” icon for the object on the right-hand side of the Outliner. Useful for complex scenes when you don’t want to assign the object to another layer. This will only work on visible layers - an object on an invisible layer will still be invisible regardless of what the Outliner says. V will toggle this property for any objects that are pre-selected in the Outliner.

Selectability

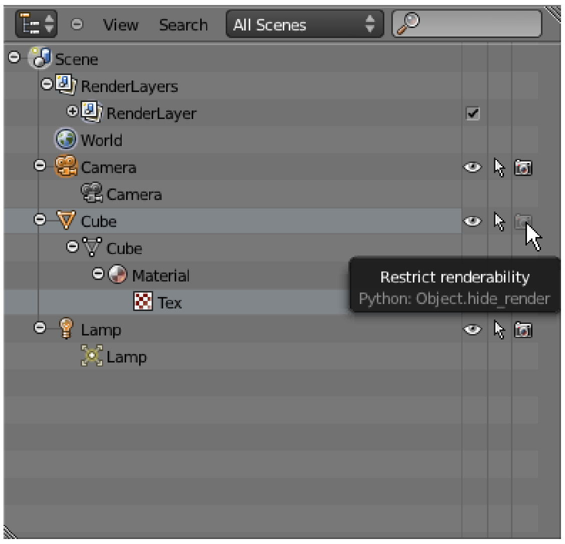


Restrict selection in the viewport
Python: Object.hide_select

Restrict selection

Toggle selectability by clicking the “arrow” icon. This is useful for if you have placed something in the scene and don’t want to accidentally select it when working on something else. S will toggle this property for any objects that are pre-selected in the Outliner.

Rendering

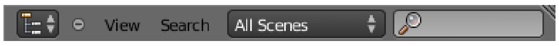


Restrict renderability

Toggle rendering by clicking the “camera” icon. This will still keep the object visible in the scene, but it will be ignored by the renderer. R will toggle this property for any objects that are pre-selected in the Outliner.

Searching

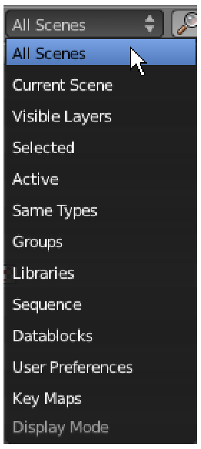
You can search the file for datablocks by (optionally) changing the parameters of the searching in the Search menu in the header of the Outliner window and then typing in the search input box in the rightmost position of this header:



The header of the Outliner window

The matching datablocks will be highlighted with the green background and nothing else except them and their parent hierarchy will be displayed.

Filtering the display

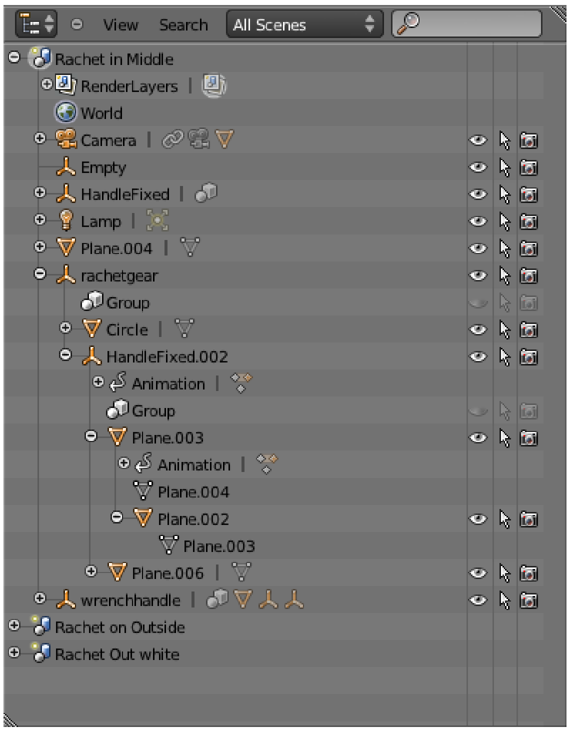


Outliner Display dropdown.

The window header has a field to let you select what the outliner should show in the outline. By default, the outliner shows All Scenes. You can select to show only the current scene, datablocks that have been selected, objects that are on currently selected layers, etc. These selects are to help you *narrowthe list* of objects so that you can find things quickly and easily.

- All Scenes - Shows *everything* the outliner can display (in all scenes, all layers, etc.)
- Current Scene - Shows everything in the current scene.
- Visible Layers - Shows everything on the visible (currently selected) layers in the current scene. Use the layers buttons to make objects on a layer visible in the 3D window.
- Selected - Lists only the object(s) currently selected in the 3D window. You can select multiple objects by ⇧ Shift RMB -clicking.
- Active - Lists only the active (often last selected) object.
- Same Types - Lists only those objects in the current scene that are of the same types as those selected in the 3d window.
- Groups - Lists only Groups and their members.
- Libraries - TODO
- Sequence - TODO
- Data Blocks - TODO
- User Preferences - TODO
- Key Maps - TODO

Example



The Outliner window in list mode.

The outline example shows that the .blend file has three scenes: “`Ratchet in Middle`”, “`Ratchet on Outside`”, and “`Ratchet Out White`”. By clicking on the little plus-sign to the left of the name, the outline is expanded one level. This was done for the “`Ratchet in Middle`” scene. As you can see, this scene has some “`World`” material settings, a “`Camera`”, an “`Empty`”, a “`HandleFixed`” object… All objects that were added to the scene.

By clicking the plus-sign next to “`ratchetgear`, we can see that it has some motion described by the “`Animation`” entry; that it was based on a “`Circle`” mesh, and that it is the parent of “`HandleFixed.002`”, which is in turn the parent of “`Plane.003`”, and so on.

The neat thing is: if you select any of these datablocks here, they will be selected in the 3D window as well, as far as this is possible. Pressing . NumPad with your mouse cursor in any 3D Window will center and align the view to that object. Very handy. Also, pressing X will delete it, as well as all the other hotkeys that operate on the currently selected object.

Linked Libraries Overview

Blender is able to “reach in” to other .blend files and pull in whatever you want. In this way, Blender supports reuse of your Blender data. For example, if you have a library .blend file that has a really neat material used in it, you can, from your current .blend file, Append that material into your current .blend file. This saves you from manually re-creating all the different settings.

General Procedure

Mode: All Modes

Hotkey: ⇧ ShiftF1

Menu: File → Append or Link

The main menu in Blender is located in the Info window (by default the header located at the top of your screen). From that menu, all you have to do is use File → Append or Link, or press ⇧ ShiftF1 in your active window. The active window will change to a File Browser (the Window type icon looks like a manila folder) selector window. Use this window to navigate your hard drive and network-mapped drives through folders and subfolders to find the .blend file that has the object you want to reuse. When you click on a .blend file (indicated by the orange square box next to its name), Blender will go into that file and show you the list of datablock types within it: Scenes, Objects, Materials, Textures, Meshes, etc. Clicking on any one of them will display the specific instances of that type.

Folder and File Organization

We suggest creating a folder called `/lib` or `/library`. Under that library, create a set of folders for each kind of thing you might want to access and re-use later on, such as `materials`, `textures` and `meshes`. Create subfolders under each of those as your library grows. For example, under the `meshes` folder, you might want to create folders for `people`, `spaceships`, `furniture`, `buildings`, etc. Then, when you have a .blend file that contains a chair mesh, for example, all you have to do is copy that file into the `furniture` folder.

Appending library objects into your current project

The following procedure appends an object with all its linked data, such as mesh data, materials, textures, …, to the current .blend file.

1. Select File → Append or Link.
2. Locate and select the file that contains the object you want to append (often a “library” file).
3. Navigate to the Object section of the file.
4. Select one object from the list using LMB, multiple objects via RMB, and/or a range of objects by dragging RMB.
5. Repeat the above for each kind of object you wish to append or link. Parents and armatures (all modifier objects) must be selected separately.
6. Set desired options that are shown in the header (At Cursor, Active Layer, …).
7. LMB on Load Library or press ↵ Enter or MMB directly on the data to append.

Of course, you can append or link many other things besides objects: all the ObData - cameras, curves, groups, lamps, materials, meshes, etc. - and even **an entire scene**… Note that there is a **big** difference between adding the object and the object data, such as mesh. If you append a Mesh datablock, you are only bringing in the data about that particular instance of mesh, and not an actual object instance of the mesh that you can see.

In the File Browser window header, use Append (button enabled by default) if you want to make a local independent copy of the object inside your file. Select Link if you want a dynamic link made to the source file; if anyone changes the object in the source file, your current file will be updated the next time you open it.

Click Load Library to append or link the object into your current .blend file.

Some more loading option buttons (in the File Browser header) include:

AutoSel
: When an object is loaded, it is not active or selected; it just plops into your .blend file. Often, right after loading, you will want to do something with it, like scale it or move it. Enable this button and the imported object will be selected, just as if you magically RMB-clicked on it. This button saves the step of finding the object and selecting it.

Active Layer
: Blender has 20 layers to divide up a large scene, and each object resides on some layer(s). By default, an object is loaded into your file directly into the layer(s) it resides on in the source file. To only load the object to the current active layer that you are working on, enable this button.

At Cursor
: By default, an object is loaded into your file at the location it is at in the source file. To reposition the object to your cursor when it loads, enable this button.

Finding What was Loaded

If the loaded object is not visible, consider using At Cursor or AutoSel. If you use AutoSel, remember there are Snap tools to put your cursor on the object (⇧ ShiftS4 (Cursor -> Selection)), and Center your view on it (C (View → Align View → Center View to Cursor)). Note that these tools do not work if the object is on an unselected layer, since objects on unselected layers are invisible.

Reusing Objects (Meshes, Curves, Cameras, Lights, …)

Let’s suppose you created a wheel in one .blend file and want to reuse it for your current project. The physical model of the wheel would be a mesh, and probably comprised of a tire and rim. Hopefully you named this mesh something reasonable, like, oh, I don’t know, “`Wheel`”. The wheel may be colored and thus have some materials assigned to it (like rubber and chrome).

Once you navigate to the file, select the “`Wheel`” (in the Objects datablocks) and it will be imported into your current file. You can import a copy of it, or merely link to it.

Linking

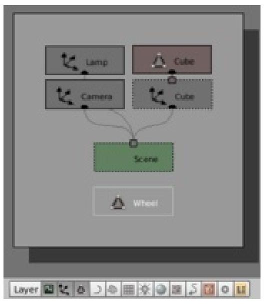
If you link to it, and later modify it in the source file, it will be shown “as-is” (modified) in your current file the next time you open it up.

Other artists have released their models to the public domain, and friends may share models simply by posting or emailing their .blend files to each other. Keeping these files, as well as your past projects, in a `Download` directory on your PC/server will save you from ever having to reinvent the wheel.

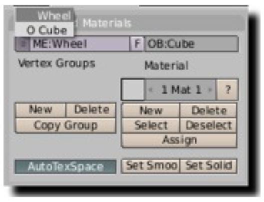
When selected, linked objects are outlined in Cyan. Normal selected objects are outlined in pink.

Notice that you cannot move a linked object! It resides at the same position it has in the source file. To move/scale/rotate the object, turn it into a proxy.

Using Appended/Linked Mesh Data



When Appending or Linking certain resources such as mesh data, it may not be instantly visible in the 3D Viewport. This is because the data has been loaded into Blender but has not been assigned to an object, which would allow it to be seen. You can verify this by looking in the Outliner window and switching it to OOPS Schematic view (you may need to have the Displays Scene datablock button selected in its header). In the OOPS Schematic picture you can see that “`Wheel`” is not linked to any object.



To allow the newly loaded `Wheel` mesh to be assigned to an object, either select a currently visible object or create a new object (such as a cube), then go to the Link and Materials panel and select the `Wheel` mesh from the mesh drop down panel, at that point you should see it, because it has been assigned to an object.

If instead of Appending/Linking to a mesh you instead load the object into Blender, it should be instantly displayed in the 3D Viewport without having to associate an object with the mesh (as it is already done!).

Reusing Material/Texture Settings

Material preview in Image Browser.

Some materials, like glass or chrome, can be very tricky to get “just right”. The Blender Foundation has released, for example, a Materials CD, which is available for free to download from their site. Using the .blend files on that CD, you can import common materials, like glass, chrome, wood and bananas. This feature saves you a lot of time, as it often means you don’t have to be fiddling with all the little buttons and sliders just to re-create a material. I call out the Banana material because it is a great example of using simple procedural materials with a ColorRamp, and a procedural texture, to give a very realistic look. When you navigate to the file, and select Materials, the browser will show you a sphere sample of that material to help you visualize the texture that goes with the name. For more information on using the Image Browser, see the release notes.

Blender Extension: Library
There is also a fantastic Python script called Blender Library that over-arches all of your files and allows you to construct a master library. This script displays a preview and helps you organize your Blender work. Highly recommended; search www.blendernation.com for “Blender Library”, it is also stored on the Blender Wiki Scripts section.

Reusing Node Layouts

To reuse noodles (node layouts), open the original (source) file and create a Group for the set of nodes that you think you want to reuse. When you want to import that node group into your current file, LMB on File → Append or LMB on File → Link from the Info window header (or press F1 for Append or CtrlAltO for Link), and navigate to the file. When you dive into the file, there will be a NodeTree option. LMB on it and the list of node groups in that file will be listed. LMB on the one you want and then LMB .

[Verse]
Verse is an amazing OpenSource collaboration tool that integrates with Blender. Verse enables multiple people to work on, link, and share objects and modifications in Blender files in real time.

Proxy Objects

A proxy is a legal stand-in or substitute for the real thing. In Blender, when you make a linked copy (described above), you cannot edit the object; all you have is a link to it. You cannot add to it or change it, because its source is in another file that is not open.

When working in a team environment, you may want more flexibility. For example, if modeling a car, you may have one person working on the shape of the car (its mesh), but another working on available color schemes (its materials). In this case, you want to grant the painter a Proxy of the object and allow him/her to modify the material settings. More commonly, you will have a character being animated by a team of animators; they can define poses, but cannot change the character’s colors or armature, only use what is defined by the master rigger.

The important aspect of a proxy object is that it allows you to edit data locally, but also allows specific data to be kept restricted. Data that’s defined as restricted will always be restored from the library (typically on file reading or undo/redo steps). This restriction is defined in the referenced library itself, which means that only the library files can define what’s allowed to change locally.

For poses, you can control this by indicating bone layers as being restricted. A restricted layer is shown with a black dot in it. Use Ctrl LMB on a button to restrict or unrestrict that layer.

Mode: Object Mode

Hotkey: CtrlAltP

To make a proxy object for yourself, establish a link to the source object as described above. With that linked copy selected (RMB) and in view (you can see it in the 3D View), press CtrlAltP and confirm the Make Proxy dialog. The object will be named with the original name plus a “`_proxy`” suffix. You may now move and modify the proxy. When selected, it will look like a local object (outlined in orange).

You can then edit unrestricted data. For most objects, this includes the location and rotation. You can also animate the object’s location using Ipo curves. For mesh objects, the shape of the mesh is restricted, so you cannot define shape keys. When you reload your file, Blender will refresh your file with any changes made to the original restricted data, but will not reset your changes (unless the owner has).

Armatures and Multiple instances

Development of this feature is a work in progress; in Blender 2.43 and CVS (as of 29 April 2007), a proxy object controls *all instances of a group*. It is not yet possible to have one proxy per group instance. In particular, it is not yet possible to have one proxy armature per group instance. One partially effective remedy to use file append rather than file link for multiple instance duplication. File append will not be updated with update to the origination file.

If you are using a POSIX compliant file system, you can work around the one proxy object per group limitation with the cheap hack documented at Linked Lib Animation Madness.

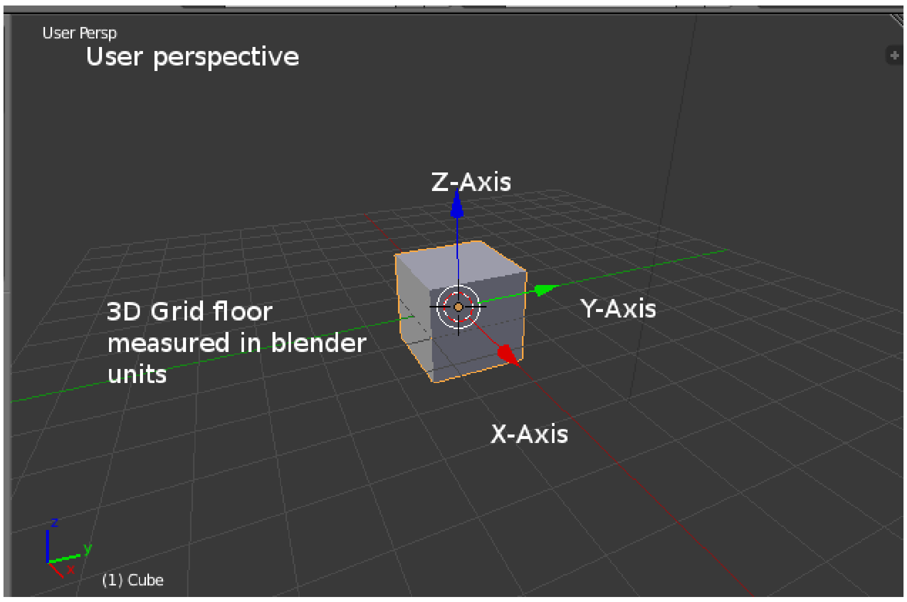
Introduction

Using Blender, you create a world that exists in four dimensions:

1. Left-right, commonly called the “*x*” axis.
2. Forward-backward, commonly called the “*y*” axis.
3. Up-down, commonly called the “*z*” axis.
4. Time-sensitive, through animated objects, materials, and motion captured in frames.

The problem is that you have a two-dimensional computer screen in front of you! Your mouse can only move left-right and up-down. You cannot go back in time, and you can’t literally reach out into the screen and grab an object and move it somewhere else.

Instead, you have to tell Blender to do it for you. This section tells you how to navigate around in your virtual world using the unique Blender interface.



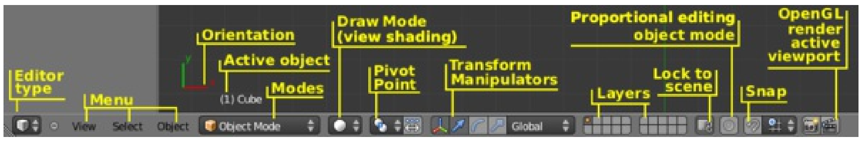
Introduction

The 3D View is where you perform most of the object modeling and scene creation. Blender has a wide array of tools and options to support you in efficiently working with your mouse, keyboard and keypad.

It is also the oldest, and therefore most feature- and option-rich area of Blender. However, there's no need to be intimidated. Just take it slow and experiment with a few options at a time to see what they do.

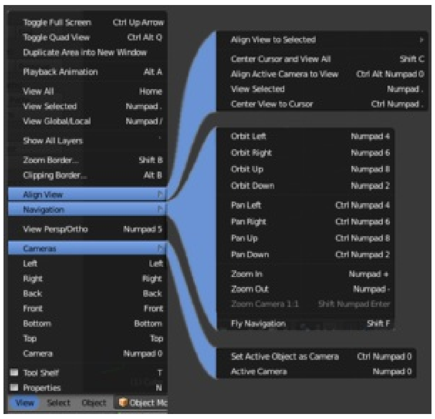
3D Window Header

The 3D View window is comprised of a workspace and a header. The header is shown at the bottom or top of the workspace, and can be hidden if desired. The header shows you a menu and the current mode, as explained below.



3D View header.

View Menu



The View menu.

Properties Panel
: Toggles the Properties side panel (N), which allows you to tweak many 3D view settings:

- Transform
- Grease Pencil
- View
- Item
- Display
- Background Images
- Transform Orientations

Tool Shelf
: Toggles the Tool Shelf (T), which appears on the left side of the 3d view, and allows you to perform various operations, depending on the type of object selected, and the mode you are in.

Camera (0 NumPad)
: Switches the view to the current camera view.

Viewing angles
: These commands change the view to the default Top/Bottom, Front/Back, or Left/Right views.

- Top (7 NumPad)
- Bottom (Ctrl7 NumPad)
- Front (1 NumPad)
- Back (Ctrl1 NumPad)
- Right (3 NumPad)
- Left (Ctrl3 NumPad)

Cameras Menu
: Set Active object as camera
Active camera

Perspective/Orthographic View(5 NumPad)
: These commands change the projection of the 3D view

Navigation Menu
: This sub-menu contains commands for rotating and panning the view. Using these commands through the menu is not that efficient. However, like all Blender menus, the much more convenient keyboard shortcuts are listed next to the commands.

Align View
: This submenu allows you to align the 3D view in certain ways.

- Align to selected
- Center cursor and view all
- Align active camera to view
- View Selected
- Center View to cursor

Clipping Border... (AltB)
: Allows you to define a clipping border to limit the 3D view display to a portion of 3D space.

Zoom Border... (⇧ ShiftB)
: Allows you to define the area you want to zoom into.

Show all Layers (~)
: Makes all of the display layers visible.

Global View/Local View (/ NumPad)
: Global view shows all of the 3D objects in the scene. Local view only displays the selected objects. This helps if there are many objects in the scene, that may be in the way. Accidentally pressing / NumPad can happen rather often if you're new to Blender, so if a bunch of the objects in your scene seem to have mysteriously vanished, try turning off local view.

View Selected (. NumPad)
: Zooms the 3D view to encompass all the *selected* objects.

Read more about Zooming the 3D View »

View All (↖ Home)
: Zooms the 3D view to encompass *all* the objects in the current scene.

Play Back Animation (AltA)
: Plays back the animation from the current frame.

Duplicate area in new window
: Clones the current 3D view in a new window

Quad View
: Toggles a four pane 3D view, each showing a different angle of the scene.

Toggle Full Screen(Ctrl↑)
: Maximizes the 3D View window to fill the full screen area.

Select Menu

This menu contains tools for selecting objects.

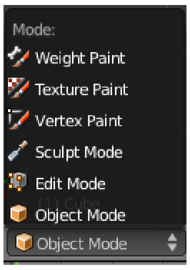
Read more about Selecting »

Object Menu

This menu appears when in Object Mode. In edit mode, it will change to the appropriate menu with editing tools.

Read more about Objects »

Mode List



The Mode drop-down list.

Blender has several modes of operation.

- Object mode allows you to work with objects as a whole.
- Edit mode by allows you to modify the shape of the object.
- Sculpt mode
    - In this mode your cursor becomes a tool to shape the object

The cursor becomes a brush in:

- Vertex Paint mode

- Weight Paint mode
- Texture Paint mode.

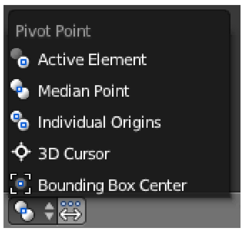
ViewPort Shading List

Allows you to change the way 3D objects are displayed in the viewport.

- Bounding Box
- Wireframe
- Solid
- Texture
- Material
- Rendered (only for Cycles renderer)

Read more about 3D view options »

Pivot Point Selector



Pivot point selector.

When rotating or scaling an object or group of vertices/edges/faces, you may want to shift the pivot point (the transformation center) in 3D space. Using this selector, you can change the pivot point to the location of the:

- Active Element
- Median Point - the average center spot of the selected items
- Individual Origins
- 3D Cursor
- Bounding Box Center

Use the Object Center to switch between transforming the entire objects, or just the position of the objects

Read more about Pivot Points »

Transform (Manipulator) Selectors

These handy selectors allow you to rotate or move objects by grabbing (clicking with your mouse) their controls and moving your mouse in the axis.

Read more about Transform Manipulators »

Layer Selector

Layers are well documented in the Layers page. Toggling layer visibility is covered in the section on viewing layers and moving objects between layers is also discussed in this page.

Lock to Scene

By default, the “lock” button to the right of the layer buttons is enabled. This means that in this view, the active layers and camera are those of the whole scene (and those used at render time). Hence, all 3D views locked this way will share the same active layers and camera – when you change them in one view, all locked others will immediately reflect these changes.

But if you disable this “lock” button, you then can specify different active layers and camera, specific to this view. This might be useful if you don’t want to have your working areas (views) cluttered with the whole scene, and still have an ancillary complete view (which is unlocked with e.g. all layers shown…). Or to have several views with different active cameras. Remember that you can use (Ctrl0 NumPad to make the active object the active camera.

Read more about Scenes »

Snap to Mesh

This “magnet” button controls the snapping tools that help with transforming and modeling objects.

Read more about Snapping »

Render Buttons

The Render Buttons render an OpenGL version of the 3D view.

The first button renders a still image of the Objects in the 3D view without displaying the grid, axes, etc. It uses the same Draw mode as the 3D view, so it’s rather useful if someone asks to see the wireframe of an Object you’re working on.

The second button will render an animation of the 3D View, making it useful for making preview renders of animations. The animation will be saved in the folder and format specified in the Output panel of the Render context.

Introduction

To be able to work in the three dimensional space that Blender uses, you must be able to change your viewpoint as well as the viewing direction of the scene. While we will describe the 3D View window, most of the other windows have similar functions. For example, it is possible to translate and zoom a Buttons window and its panels.

Mouse Buttons and Numpad

If you have a mouse with less than three buttons or a keyboard without numpad, please refer to the Keyboard and Mouse page of the manual to learn how to use them with Blender.

Perspective and Orthographic Views

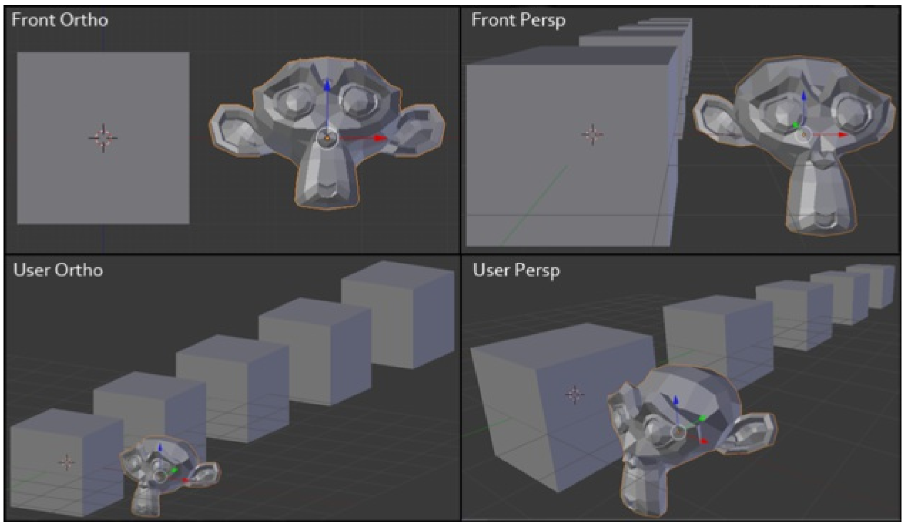
Mode: All modes

Hotkey: 5 NumPad

Menu: View » Perspective / View » Orthographic

Description

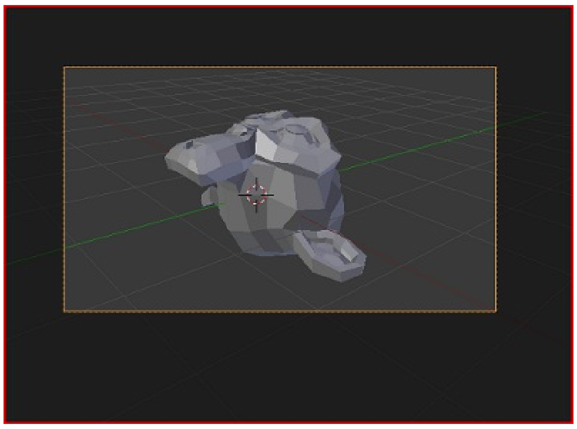
Each 3D viewport supports two different types of projection. These are demonstrated in the *Orthographic (left) and perspective (right) projections* image below.



Orthographic (left) and perspective (right) projections.

Our eye is used to perspective viewing because distant objects appear smaller. Orthographic projection often seems a bit odd at first, because objects stay the same size regardless of their distance. It is like viewing the scene from an infinitely distant point. Nevertheless, orthographic viewing is very useful (it is the default in Blender and most other 3D applications), because it provides a more “technical” insight into the scene, making it easier to draw and judge proportions.

Options


Demonstration of camera view.

To change the projection for a 3D view, choose the View » Orthographic or the View » Perspective menu entry. The 5 NumPad shortcut toggles between the two modes. Changing the projection for a 3D view does not affect the way the scene will be rendered. Rendering is in perspective by default. If you need to create an orthographic rendering, select the camera, go to the Object Data context and press the Orthographic button in the Lens panel.

The View » Camera menu entry sets the 3D view to camera mode (0 NumPad). The scene is then displayed as it will be rendered later (see *Demonstration of camera view*). The rendered image will contain everything within the orange dotted line. Zooming in and out is possible in this view, but to change the viewpoint, you have to move or rotate the camera.

If you have a large scene, viewing it through Camera View may not display all of the Objects in the scene. One possibility may be that the clipping distance of the camera is too low. The camera will only show objects that fall within the clipping range.

Read more about Render perspectives »
Read more about Camera View »
Read more about Camera clipping »

Technical Details

Perspective definition

A *perspective* view is geometrically constructed by taking a scene in 3D and placing an observer at point `O`. The 2D perspective scene is built by placing a plane (e.g. a sheet of paper) where the 2D scene is to be drawn in front of point `O`, perpendicular to the viewing direction. For each point `P` in the 3D scene a `PO` line is drawn, passing by `O` and `P`. The intersection point `S` between this `PO` line and the plane is the perspective projection of that point. By projecting all points `P` of the scene you get a perspective view.

Orthographic definition

In an *orthographic* projection, you have a viewing direction but not a viewing point `O`. The line is then drawn through point `P` so that it is parallel to the viewing direction. The intersection `S` between the line and the plane is the orthographic projection of the point `P`. By projecting all points `P` of the scene you get the orthographic view.

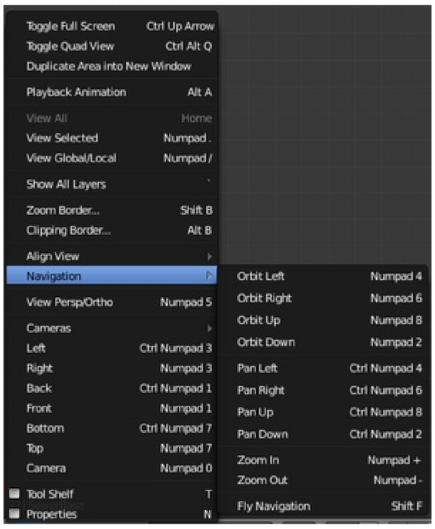
Rotating the View

Mode: All modes

Hotkey: MMB / 2 NumPad / 4 NumPad / 6 NumPad / 8 NumPad / CtrlAlt Wheel

Menu: View » Navigation

Description



A 3D viewport's View menu.

Blender provides four default viewing directions: Side, Front, Top and Camera view. Blender uses a right-angled "Cartesian" coordinate system with the Z axis pointing upwards. "Side" corresponds to looking along the X axis, in the negative direction, "Front" along the Y axis, and "top" along the Z axis. The Camera view shows the current scene as seen from the camera view point.

Options

You can select the viewing direction for a 3D viewport with the View menu entries, or by pressing the hotkeys 3 NumPad for "side", 1 NumPad for "front", 7 NumPad for "top". You can select the opposite directions if you hold Ctrl while using the same numpad shortcuts. Finally 0 NumPad gives access to the "camera" viewpoint.

Apart from these four default directions, the view can be rotated to any angle you wish. Click and drag MMB on the viewport's area. If you start in the middle of the window and move up and down or left and right, the view is rotated around the middle of the window. Alternatively, if the Emulate 3 button mouse option is select in the User Preferences you can press and hold Alt while dragging LMB in the viewport's area.

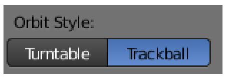
To change the viewing angle in discrete steps, use 8 NumPad and 2 NumPad (which correspond to vertical MMB dragging, from any viewpoint), or use 4 NumPad and 6 NumPad (or CtrlAlt Wheel) to rotate the scene around the Z global axis from your current point of view.

Hotkeys
Remember that most hotkeys affect **the active window** (the one that has focus), so check that the mouse cursor is in the area you want to work in before your use the hotkeys.

TrackBall/Turntable

By default, when you rotate the view as described above, you are using the **turntable** method. For some users this is intuitive and for others it is not. If you feel you are having difficulties with this style of 3D window rotation you can switch to the "**trackball**" style. With the trackball style you are rotating the scene as though you are rolling your hand across a "**trackball**"

The Turntable style is fashioned more like a record player where you have two axes of rotation available, and the world seems to have a better definition of what is "Up" and "Down" in it. The downside to using the Turntable style is that you lose some flexibility when working with your objects. However, you gain the sense of "Up" and "Down" which can help if you are feeling disoriented. Of course you can always switch between the styles depending on what you are working on.



View rotation.

To change the rotation "style", use the User Preferences window. Click on the Input button and you will see an option for choosing the Orbit style. There are two additional checkboxes for controlling the display in the 3D window in the Interface tab in the User Preferences. Auto Perspective will automatically switch to perspective whenever the view is rotated using MMB. Rotate Around Selection will rotate the view around the center of the current selection. If there is no selection at that moment (e.g. if you used A to deselect everything), the last selection will be used anyway.

Panning the View

Mode: All modes

Hotkey: ⇧ Shift MMB / Ctrl2 NumPad / Ctrl4 NumPad / Ctrl6 NumPad / Ctrl8 NumPad / ⇧ ShiftAlt LMB

Menu: View → Navigation

Description

To pan the view, hold down ⇧ Shift and drag MMB in the 3D Viewport. For discrete steps, use the hotkeys Ctrl8 NumPad, Ctrl2 NumPad, Ctrl4 NumPad and Ctrl6 NumPad as with rotating (note: you can replace Ctrl by ⇧ Shift). For those without a middle mouse button, you can hold ⇧ Shift Alt while dragging with LMB.

Zooming the View

Mode: All modes

Hotkey: Ctrl MMB / Wheel / + NumPad / - NumPad

Menu: View → Navigation

Description

You can zoom in and out by holding down Ctrl and dragging MMB. The hotkeys are + NumPad and - NumPad. The View » Navigation sub-menu holds these functions too as well. Refer to the 3D viewport's View menu image above for more information.

If you have a wheel mouse, you can perform all of the actions in the 3D viewport that you would do with + NumPad and - NumPad by rotating the Wheel. To zoom a Buttons window, hold Ctrl MMB and move your mouse up and down.

If You Get Lost…
If you get lost in 3D space, which is not uncommon, two hotkeys will help you: ↖ Home changes the view so that you can see all objects (View » View All menu entry), while . NumPad zooms the view to the currently selected objects when in perspective mode (View » View Selected menu entry).

Zoom Border

The Zoom Border tool allows you to specify a rectangular region and zoom in so that the region fills the 3d view.

You can access this through the View menu, or the shortcut ⇧ ShiftB then click and drag rectangle to zoom in.

Dolly the View

Mode: All modes

Hotkey: Ctrl⇧ Shift MMB

Description

In most cases its sufficient to zoom the view to get a closer look at something, however you may notice that at a certain point you cannot zoom any closer.

This is because Blender stores a view-point thats used for orbiting and zooming, This works well in many cases but sometimes you want to move the view-point to a different place - This is what Dolly supports, allowing you to transport the view from one place to another.

You can dolly back and fourth by holding down Ctrl⇧ Shift and dragging MMB.

Aligning the View

Align View

These options allow you to align and orient the view in different ways. They are found in the View Menu

Align View to Selected menu

These options align your view with specified local axes of the selected object or, in Edit mode, with the normal of the selected face.

Top ⇧ Shift7 NumPad
Bottom ⇧ ShiftCtrl7 NumPad
Front ⇧ Shift1 NumPad
Back ⇧ ShiftCtrl1 NumPad
Right ⇧ Shift3 NumPad
Left ⇧ ShiftCtrl3 NumPad

Center Cursor and View All (⇧ ShiftC)

moves the cursor back to the origin **and** zooms in/out so that you can see everything in your scene.

Align Active Camera to View, CtrlAlt0 NumPad

Gives your active camera the current viewpoint

View selected, . NumPad

Focuses view on currently selected object/s by centering them in the viewport, and zooming in until they fill the screen.

Center view to cursor, Alt↖ Home

Centers view to 3D-cursor

View Selected

See above

View All ↖ Home

Frames all the objects in the scene, so they are visible in the viewport.

Local and Global View

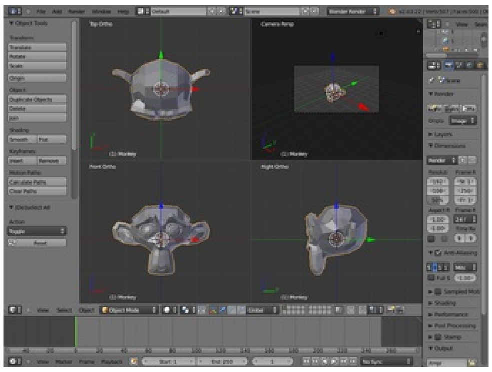
You can toggle between Local and Global view by selecting the option from the View Menu or using the shortcut / NumPad. Local view isolates the selected object or objects, so that they are the only ones visible in the viewport. This is useful for working on objects that are obscured by other ones, or have heavy geometry. Press / NumPad to return to Global View.

Quad View

Mode: All modes

Hotkey: CtrlAltQ

Menu: View » Toggle Quad View


Quad View

Toggling Quad View will split the 3D window into 4 views: Top Ortho, Front Ortho, Right Ortho and Camera / User View. This view will allow you to instantly see your model from a number of view points. In this arrangement, you can zoom and pan each view independently but you cannot rotate the view. Note that this is different from splitting the windows and aligning the view manually. In Quad View, the four views are still part of a single 3D window. If you want to be able to rotate each view, you will need to split the 3D window into separate windows.

Read more about splitting windows »

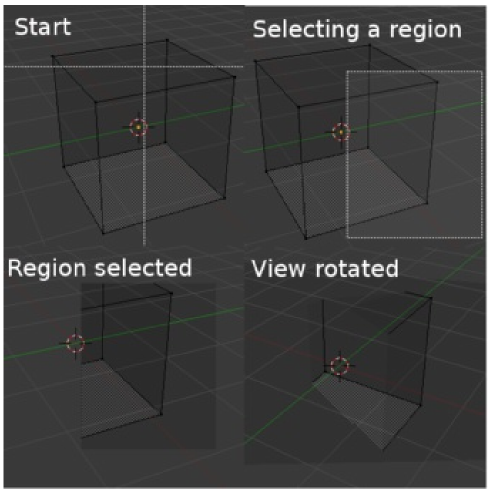
View Clipping Border

Mode: All modes

Hotkey: AltB

Menu: View » Set Clipping Border

Description



Region/Volume clipping.

To assist in the process of working with complex models and scenes, you can set the view clipping to visually isolate what you’re working on.

Once clipping is used, you will only see whats inside a volume you've defined. Tools such as paint, sculpt, selection, transform-snapping etc. will also ignore geometry outside the clipping bounds.

Once activated with AltB, you have to draw a rectangle with the mouse, in the wanted 3D view. The created clipping volume will then be:

- A right-angled parallelepiped (of infinite length) if your view is orthographic.
- A rectangular-based pyramid (of infinite height) if your view is in perspective.

To delete this clipping, press AltB again.

Example

The *Region/Volume clipping* image shows an example of using the clipping tool with a cube. Start by activating the tool with AltB (upper left of the image). This will generate a dashed cross-hair cursor. Click with the LMB and drag out a rectangular region shown in the upper right. Now a region is defined and clipping is applied against that region in 3D space. Notice that part of the cube is now invisible or clipped. Use the MMB to rotate the view and you will see that only what is inside the pyramidal volume is visible. All the editing tools still function as normal but only within the pyramidal clipping volume.

The dark gray area is the clipping volume itself. Once clipping is deactivated with another AltB, all of 3D space will become visible again.

View Navigation

Mode: All modes

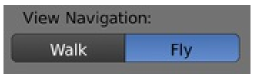
Hotkey: ⇧ ShiftF

Description

When you have to place the view, normally you do as described above.

However, there are cases in which you really prefer to just navigate your model, especially if it’s very large, like environments or some architectural model. In these cases viewing the model in perspective mode has limitations, for example after zooming a lot of panning is extremely uncomfortable and difficult, or you apparently cannot move the camera any nearer. As an example, try to navigate to a very distant object in the view with traditional methods (explained above) and see what you can get.

With Walk mode and Fly mode you move, pan and tilt, and dolly the camera around without any of those limitations.



In the User Preferences window select the navigation mode you want to use as default when invoking the View Navigation operator. Alternatively you can call the individual modes from the View Navigation menu.

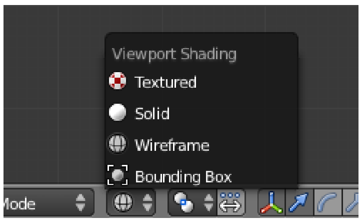
View Shading

Mode: All modes

Hotkey: Z / ⇧ ShiftZ / AltZ / ⇧ ShiftAltZ / D

Description

Depending on the speed of your computer, the complexity of your scene, and the type of work you are currently doing, you can switch between several drawing modes:



A 3D view's draw mode button.

Textured
: Displays UV image textured models with OpenGL lighting. Neither procedural textures or non UV-mapped textures will be shown.

Shaded
: Approximates all textures and lighting at each vertex, and blends from one to the next. Much less accurate than using the render engine to check textures, but much faster. Note that if you have no lighting in your scene, everything will remain black.

Solid
: This is the default drawing mode where surfaces are drawn as solid colors, with built-in OpenGL lighting. This draw mode is not dependent on scene light sources and can be configured in the Solid OpenGL lights group of controls from the System & OpenGL tab of the User Preferences window.

  Read more about System Configuration »

Wireframe
: Objects only consist of lines that make their shapes recognizable (e.g. the edges of meshes or surfaces…).

Bounding Box
: Objects aren't drawn at all. Instead, this mode shows only the rectangular boxes that correspond to each object's size and shape.

You can switch between these draw modes by:

- Using the Draw type drop-down list in the 3D views' header (see *A 3D view's draw mode button*).
- Pressing D to pop-up the Draw mode menu.
- Using the Z-based shortcuts as detailed below:

Draw modes and Z-based shortcuts.

Z	Switches between Wireframe and Solid draw modes.
⇧ ShiftZ	Switches between Wireframe and Shaded draw modes.
AltZ	Switches between Solid and Textured draw modes.
⇧ ShiftAltZ	Switches to the Textured draw mode.

View Properties Panel

Mode: All modes

Panel: View Properties

Menu: View » View Properties...

Description

In addition to the header controls described above, the View Properties panel lets you set other settings regarding the 3D view. You show it with the View » View Properties... menu entry.

View

Lens
: Control the focal length of the 3d view camera in millimeters, unlike a rendering camera

Lock to Object
: By entering the name of an object in the Object field, you lock your view to this object, i.e. it will always be at the center of the view (the only exception is the active camera view, 0 NumPad).
  If the locked object is an armature, you can further center the view on one of its bones by entering its name in the Bone field.

Lock to Cursor
: Lock the center of the view to the position of the 3D cursor

Lock Camera to View
: When in camera view, use this option to move the camera in 3D space, while continuing to remain in camera view.

Clip Start and Clip End
: Adjust the minimum and maximum distances to be visible for the view-port.

Notice

A large clipping range will allow you to see both near and far objects, but reduces the depth precision.

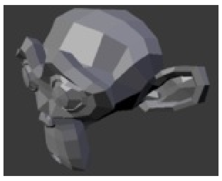
To avoid this...

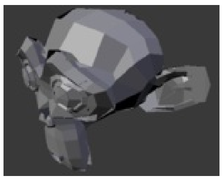
- increase the near clipping when working on large scenes.
- decrease the far clipping when objects are not viewed at a distance.

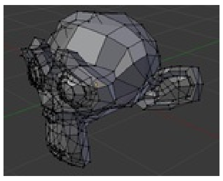
When perspective is disabled only the far Clip-End is used, very high values can still give artifacts.

This is not specific to blender, all OpenGL/DirectX graphics applications have these same limitations.

Examples:


Model with no clipping artifacts.


Model with clipping artifacts.


Mesh with artifacts in edit-mode.

Local Camera
: Active camera used in this view

3D Cursor Location
: Here you can precisely specify the position of the 3D cursor

Item

This section displays the currently selected object

Display

Only Render
: Displays only items that will be rendered.

Outline Selected
: If disabled, the pink outline around your selected objects in Solid/Shaded/Textured draw types will no longer be displayed.

All Object Origins
: If enabled, the center dot of objects will always be visible, even for non-selected ones (by default, unselected centers might be hidden by geometry in solid/shaded/textured shadings…).

Relationship Lines
: Controls whether the dashed parenting, constraining, hooking, etc., lines are drawn.

All Edges
: When wire overlay is enabled in the Object context, this options forces all of the wireframe to be displayed in the viewport.

Grid Floor
: If disabled, you have no grid in other views than the orthographic top/front/side ones.

X Axis, Y Axis, Z Axis
: Control which axes are shown in other views than the orthographic top/front/side ones.

Lines
: Controls the number of lines that make the grid in non-top/front/side orthographic views, in both directions.

Scale
: Control the scale of the grid floor

Subdivisions
: Controls the number of sub-lines that appear in each cell of the grid when you zoom in, so it is a setting specific to top/front/side orthographic views.

Shading
: Control the way objects in the 3D view are shaded.

Textured Solid
: Display face assigned textures in solid view.

Toggle Quad View
: Toggles the four pane 3D view. Read more about arranging frames »

Background Image

Mode: All modes

Panel: Background Image

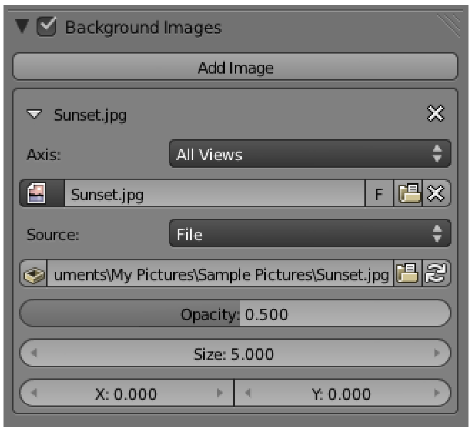
Menu: View » Properties...

A background picture in your 3D view is very helpful in many situations: modeling is obviously one, but it is also useful when painting (e.g. you can have reference pictures of faces when painting textures directly on your model…), or animation (when using a video as background), etc.

There are a few points worth to be noted about background images:

- They are specific to their window (i.e. you can have different backgrounds for each of your 3D views, e.g. top/front/side images for relevant views…).
- *They are only available for Top, Side and Front (and their complementary versions) orthographic views!* The picture remains the same when you switch between these six views.
- Their size is related to the window's zooming factor (i.e. they grow big when you zoom in, etc.).
- You can use video files and animated sequences.

Settings



The Background Image panel.

Blender manages this feature through the Background Image menu on the view properties panel (N). The option box at the top of this panel toggles the Background Image feature on/off. By default, there is only space for one image. The settings can be accessed by LMB the white triangle.

Once enabled, you can add an image by selecting an existing datablock, or loading a new image. The Axis menu defines which views the image will appear in. Additional images can be added by LMB the Add Image button. When the image is loaded, the following settings become available.

Source
: Specifies what type of file is being used. Depending on the selected type, several options will appear below:

  File

    Use an image file

    Source File

      Represents the actual file that is linked to the current datablock. Supported formats include bmp, gif, jpg, png, tga, and tif.

  Sequence

    a sequence of numbered image files

    Frames

      Set the number of image files to use in the sequence

    Start

      Sets the frame number to start on

    Offset

      Offsets the number of the frame used in the sequence

    Fields

      Sets the number of fields per rendered frame

Auto Refresh

Always refresh the image on frame changes

Cyclic

Cycle the images in the sequence

Movie

Use a movie file:

Match Movie Length

Set the number of frames to match the movie

Generated

Use a image generated in Blender:

Width, Height

Set the width and height if the image in pixels

Blank

Generates a blank image

UV Grid

Creates a grid for testing UV mappings

Color Grid

Creates a colored grid for testing UV mappings

Opacity

This slider controls the transparency of the background image (from **0.0** – fully opaque – to **1.0** – fully transparent).

Size

Controls the size, or scale, of the picture in the 3D view (in Blender units). This is a scalar value so that width and height of the background image are each multiplied by the value to determine the size at which the background image is displayed. If one wishes to change the proportions of the image, it must be done in an impage processing program, such as GIMP.

X Offset, Y Offset

The horizontal and vertical offset of the background image in the view (by default, it is centered on the origin), in Blender units.

Use Lo-Res Proxy

To improve PC performance when using background images you may have to use lower-resolution proxies. If your monitor resolution is 800×600, then the background image, full screen, without zooming, only needs to be 800×600. If your reference image is 2048×2048, then your computer is grinding away throwing away pixels. Try instead to take that 2k×2k image, and scale it down (using Blender, or Gimp) to, for example, 512×512. You will have sixteen times the performance, with no appreciable loss of quality or exactness. Then, as you refine your model, you can increase the resolution.

Shortcuts

Page status (reviewing guidelines)

Void page
Proposed fixes: deletion, redundant content.

Navigation Modes

Walk and fly mode allow you to interactively navigate the scene.

This mode actually *moves* the camera used by the view. This means that *when you are in Camera view(0 NumPad), it moves the active camera*, which is another way to place and aim it. In other views (top, front, user, …), it moves the view's virtual camera…

Walk mode

Mode: All modes

Menu: View » Navigation » Walk Navigation

Description

This navigation mode behaves similar to the first person navigation system available in most 3d world games nowadays. It works with a combination of keyboard keys (WASD) and mouse movement. By default the navigation is in the 'free' mode, with no gravity influence. You can toggle between gravity and free mode during the navigation (⇆ Tab).

Options

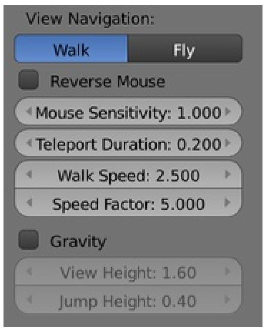
To enter Walk mode, in Object or Edit mode, set it as default in the User Preferences window and press ⇧ ShiftF: the mouse pointer will move at the center of the view, and a cross marker will appear… In any other mode, you can access it *via* View » View Navigation » Camera Walk Mode.

To move to places more quickly you can teleport (Space) around your scene. If there is an object in front of the walk cross/aim you will move in that direction until you reach the point (offset by the 'camera height' value set in the [Doc:2.6/Manual/Preferences|User Preferences window]]).

You can:

- Move the mouse left/right to pan the view left/right or move it up/down to tilt the view up/down.
- Move the camera forward/backward (W/S).
- Strafe left/right (A/D).
- Jump (V) - only in 'gravity' mode.
- Move up and down (Q/E) - only in 'free' mode.
- Alternate between 'free' and 'gravity' modes (⇆ Tab).
- Change the movement speed:
  - WheelUp or + NumPad to increase the movement speed for this open session
  - WheelDown or to - NumPad to decrease the movement speed for this open session
  - ⇧ Shift (hold) - to speed up the movement temporarily
  - Alt (hold) - to slow down the movement temporarily

When you are happy with the new view, click LMB to confirm. In case you want to go back from where you started, press Esc or RMB, as usual.


View Navigation:
Walk [✓] Fly
Reverse Mouse
Mouse Sensitivity: 1.000
Teleport Duration: 0.200
Walk Speed: 2.500
Speed Factor: 5.000
Gravity
View Height: 1.60
Jump Height: 0.40

If the defaults values (speed, mouse sensitivity, ...) need adjustments for your project, in the User Preferences window you can select a few options for the navigation system:

Fly mode

Mode: All modes

Menu: View » Navigation » Fly Navigation

Description

Fly mode lets you navigate the view using momentum set by the mouse and keys.

Options

To enter Fly mode, in Object or Edit mode, set it as default in the User Preferences window and press ⇧ ShiftF: the mouse pointer will move at the center of the view, and a squared marker will appear – a sort of HUD… In any other mode, you can access it *via* View » View Navigation » Camera Fly Mode.

Some of the options of Fly mode are influenced by the position of the mouse pointer relative to the center of the view itself, and the squared marker around this center provides a sort of “safe region” where you can place the mouse for it to have no effect on the view. The more you take the mouse pointer away from the marker, the more you pan, or track, etc.

You can:

- Move the mouse left/right to pan the view left/right or move it up/down to tilt the view up/down.
- Move the view forward/backward:
  - WheelUp or + NumPad to accelerate the movement forward.
  - WheelDown or to - NumPad to accelerate the movement backward.

  So if the view is already moving forward, WheelDown /- NumPad will eventually stop it and then move it backward, etc.
- Drag the MMB to pan. In this case the view can move laterally on its local axis at the moment you drag the mouse – quite obviously, dragging left/right/up/down makes the view point move in this direction without rotating.

When you are happy with the new view, click LMB to confirm. In case you want to go back from where you started, press Esc or RMB , as usual.

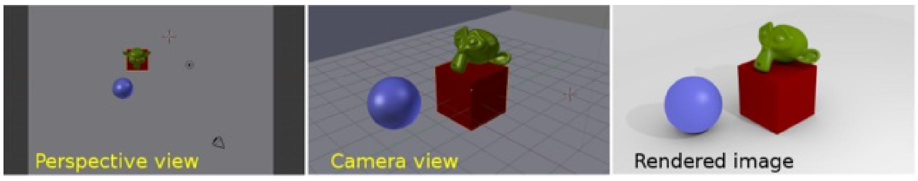
Camera View

Mode: All modes

Hotkey: 0 NumPad

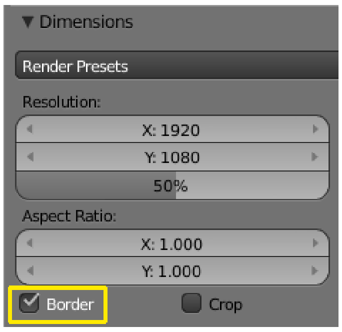
Menu: View » Camera » Active Camera

Cameras View can be used to virtually compose shots and preview how the scene will look when rendered. Pressing 0 NumPad will show the scene as viewed from the currently active camera. In this view you can also set the Render Border which defines the portion of the camera view to be rendered.



Camera view provides a preview for the final rendered image.

Render Border


▼ Dimensions

Render Presets

Resolution:

X: 1920

Y: 1080

50%

Aspect Ratio:

X: 1.000

Y: 1.000

☑ Border ☐ Crop

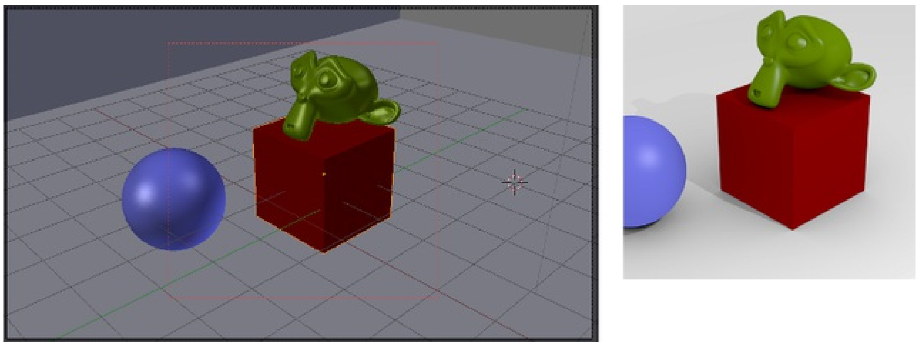
Render Border toggle

While in camera view, you can define a Render Border by pressing CtrlB. This will allow you to draw out a dotted orange rectangle within the camera view. Your renders will now be limited to the part of scene visible within the render border. This can be very useful for reducing render times. The border can be disabled by disabling the Border option in the Dimensions panel of the Render context or by using CtrlB to set a Render Border larger than the camera view.

Anti-Aliasing and blur options with borders
Note that when Render Borders are activated, Full Sampling Anti-Aliasing will be disabled while Sampled Motion Blur will become available.

Read more about Anti-Aliasing »

Read more about Motion Blur »


Render border and associated render.

Read more about Render Output options »

Read more about Cameras »

Layers

Mode: Object mode

Panel: Relations (Object context)

Hotkey: M

Menu: Object » Move to Layer...

3D scenes often become exponentially more confusing as they grow more complex. Sometimes the artist also needs precise control over how individual objects are lit, and does not want lights for one object to affect nearby objects. For this and other reasons below, objects can be placed into one or more “layers”. Using object layers, you can:

- Selectively display objects from certain layers in your 3D view, by selecting those layers in the 3D View header bar. This allows you to speed up interface redrawing, reduce virtual-world clutter, and help improve your workflow.
- Control which lights illuminate an object, by making a light illuminate only the objects on its own layer(s).
- Control which forces affect which particle systems, since particles are only affected by forces and effects on the same layer.
- Control which layers are rendered (and hence, which objects), and which properties/channels are made available for compositing by using render layers.

Armatures can also become very complex, with different types of bones, controllers, solvers, custom shapes, and so on. Since armatures are usually located close together, this can quickly become cluttered. Therefore, Blender also provides layers just for armatures. Armature layers are very similar to object layers, in that you can divide up an armature (rig) across layers and only display those layers you wish to work on.

Read more about armature layers »

Working with Layers

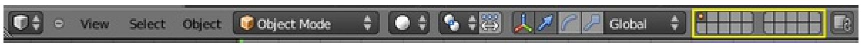
3D layers differ from the layers you may know from 2D graphics applications as they have no influence on the drawing order and are there (except for the special functions listed above) mainly to allow you to organize your scene.

When rendering, Blender only renders the selected layers. If all your lights are on a layer that is *not selected*, you won’t see anything in your render except for objects lit by ambient lighting.

Groups and Parenting are other ways to logically group related sets of objects. Please refer to the relevant sections for more information.

Viewing layers

Blender provides twenty layers whose visibility can be toggled with the small unlabeled buttons in the header (see *3D Viewport layer buttons*). To select a single layer, click the appropriate button with LMB; to select more than one, use ⇧ Shift LMB – doing this on an already active layer will deselect it.



3D Viewport layer buttons.

To select layers via the keyboard, press 1 to 0 (on the main area of the keyboard) for layers 1 through 10 (the top row of buttons), and Alt1 to Alt0 for layers 11 through 20 (the bottom row). The ⇧ Shift key for multiple (de)selection works for these shortcuts too.

You can select or deselect all Scene Layer buttons at once by pressing the ` key.

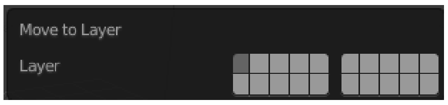
Locking to the scene

By default, the lock button directly to the right of the layer buttons is enabled. This means that changes to the viewed layers affect all other 3D Views locked to the scene – see the navigating the 3D view options page for more information.

Multiple Layers

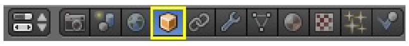
An object can exist on multiple layers. For example, a lamp that only lights objects on a shared layer could “be” on layers 1, 2, and 3. An object on layers 3 and 4 would be lit, whereas an object on layers 4 and 5 would not. There are many places where layer-specific effects come into play, especially lights and particles.

Moving objects between layers

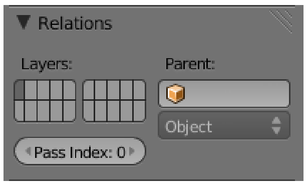


Layer selection.

To move selected objects to a different layer, press M and then select the layer you want from the pop-up dialog. Objects can also be on more than one layer at a time. To have an object on multiple layers, hold ⇧ Shift while clicking.


Object context selection.

Another way to view or change a selected object layer is via the Relations panel, in the Object context.



Layers in Object context,
Relations panel.

You will then see the layer buttons in the Relations panel – as before the object can be displayed on more than one layer by clicking ⇧ Shift LMB.

Example of object layer arrangement

As a suggestion, use the top row of layers for important parts of your scene, and the bottom row for those you don’t use or change often (or for alternatives for the top row). In a staged set involving mainly two actors, you might have the following objects on your layers:

1. Lead Actors.
2. Supporting Actors.
3. Supporting Crew (background actors).
4. Particles and effects (vortex, wind).
5. Main Stage.
6. Main backdrops and panels.
7. Main props (tables, chairs).
8. Little props, fillers, decorations, trappings.
9. Cameras, Lights.
10. Lead Actors’ armatures.
11. Supporting Actors’ armatures.
12. Crew armatures.
13. Alternative clothing.
14. Mesh WIP.
15. Different stage setup, dimensions.
16. Different backdrops that could be used.
17. Other big props that might clog up the scene.
18. Props WIP.
19. Additional lighting.

Local or Global View

Mode: All modes

Hotkey: Numpad/

Menu: View » Local View or View » Global View

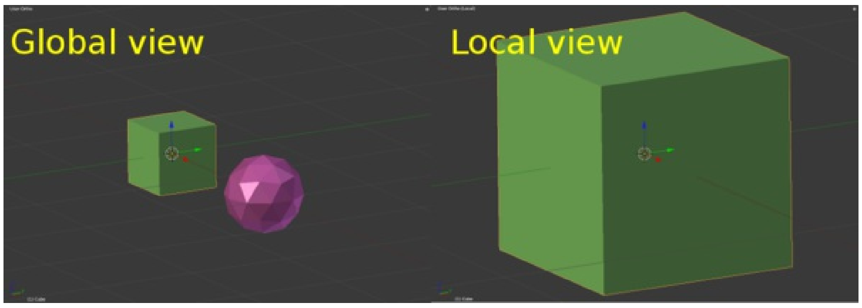
Description

When in Local view, only the selected objects are displayed, which can make editing easier in complex scenes. To enter local view, first select the objects you want, and then use the View » Local View menu entry. Use the View » Global View menu entry to go back to global view. Numpad/ toggles between both views.

Note that the layer and lock buttons on the 3D View header disappear while in local view.

Examples

In *Global view*, all the objects are visible. With the green cube selected, switching to *Local view* with Numpad/ will center the cube in the 3D View. If a scene has thousands of objects visible, this feature can potentially speed interactivity up because only the objects you selected will be visible.

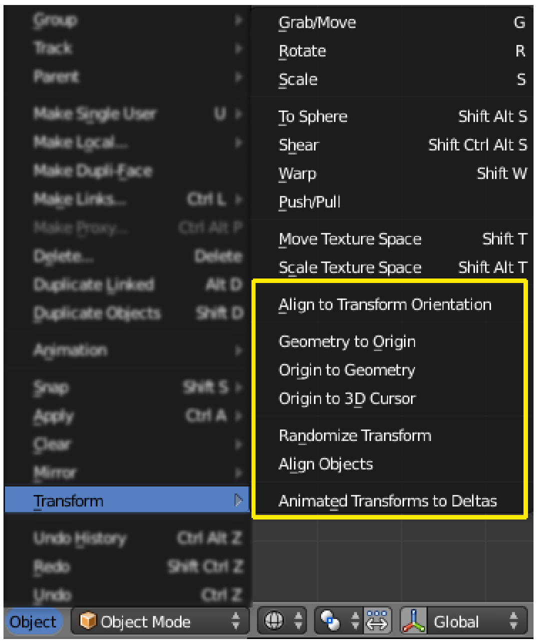


Global and Local view

Transformations

Mode: Object and Edit Mode

Menu: Object/Mesh » Transform



Transform menu in Object Mode. The yellow highlighted sections are only available in Object Mode.

Transformations refer to a number of operations that can be performed on a selected Object or Mesh that alters its position or characteristics. Basic transformations include grabbing (moving), rotating or scaling a selection. More advanced transformations included mirroring, giving the selection sphere like qualities, shear, push/pulling and warping. The following links provide a more detailed explanation of the more available transformation operations.

Basic transformations

- Grab/Move: move a selection.
- Rotate: rotate a selection.
- Scale: change the size of a selection.

Advanced Transformations

- Mirror: mirror the selection.
- To Sphere: make the selection have a more spherical shape.
- Shear: shear the selection. Shearing causes parallel selections to move past one another.
- Warp: warp the selection.
- Shrink/Fatten: Move vertices along their normals (Mesh Editmode only).
- Push/Pull: push or pull the selection (imagine someone pushing or pulling at the ends of the selection to stretch or compress it).
- Move Texture Space: Texture space determines the placement of textures. Moving it can be useful when mapping textures.
- Scale Texture Space: As above. Useful when mapping textures.
- Align to Transform Orientation: Aligns the Object to the current Transform Orientation.
- Geometry to Origin: Move the Object's geometry to the origin point.
- Origin to Geometry: Move the Object's origin to its geometry.
- Origin to 3D cursor: Move the Object's origin to the 3D cursor.
- Randomize Transform: Apply random movement, rotation and scale to selected Objects.
- Align Objects: Align Objects along a particular axis.
- Animated Transforms to Deltas: Converts animated Transform values to Delta Transform values. Allows duplicated Objects with keyframes to have offsets (location, rotation, scale etc).

Transform Control

In addition to the specific controls on each of the above pages, there are a number of general controls that can be used to modify the effects of the listed transformations. This includes using keyboard input for precise control, resetting transformations and axis locking.

Read more about Transform Controls »

Basic Manipulations

This section gathers all basic transformations and manipulations in 3D Views:

- Grab (move) elements.
- Rotate elements.
- Scale elements.
- Snap elements to something (geometry, grid, etc.) during transformations.
- How to control the precision of the transformations.
- Make transformations by typing numbers rather than moving the mouse.
- Reset object’s transformations.

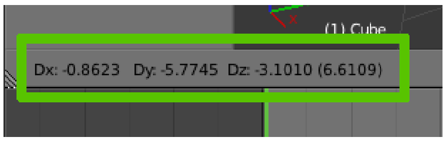
Grab/Move

Mode: Object Mode, Edit Mode, and Pose Mode for the 3D View; UV/Image Editor Tools, Sequence Editor, Dopesheet, and Graph Editor for other specific types of Grab/Move operations.

Hotkey: G or combinations for specific Axis constraint

Menu: Context Sensitive, Object Based → Transform → Grab/Move

This option lets you translate your Objects when in Object Mode, or the elements that are used in the Object construction in the 3D space of the active 3D Viewport.It offers similar functionality in various other editor environments like Node editor, Graph editor, UV editor, Sequencer etc. Further details of the option will be discussed in the related sections in detail.


Dx: -0.8623 Dy: -5.7745 Dz: -3.1010 (6.6109)

Translation Display

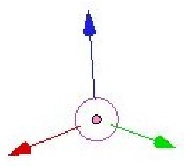
While translating, the amount of change in the co-ordinates is displayed at the bottom left corner of the 3D view window.

3D View

There are **2** types of Grab/Move options in the 3D View:

- Using shortcuts and combinations of shortcuts.
- Using the Transform Widget helper, when you choose the Translation Widget in the header of the 3DView.

Transform Widget



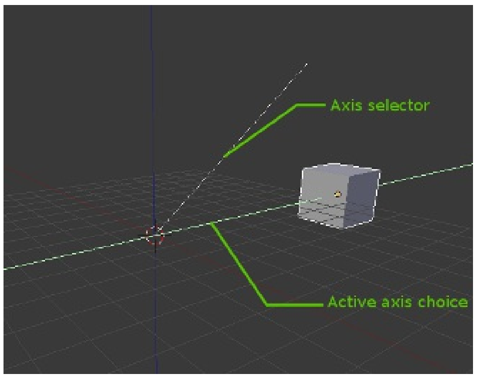
Translation Widget

In the default installation of Blender, this is the Transform Widget active by default. You can access this option by click holding LMB and dragging the 3D translatation widget in the 3D view itself.

Shortcuts in the 3D View

One of the fastest ways to move things in 3D space is with G. Pressing this hotkey will enter the “grab/move” transformation mode, where the selected object or data is moved freely, according to the mouse pointer’s location. Using combinations of this shortcut with specifc shortcuts to specify a chosen axis, will give you full control over your transformation

LMB
: Confirm the move, and leave the object or data at its current location on the screen.



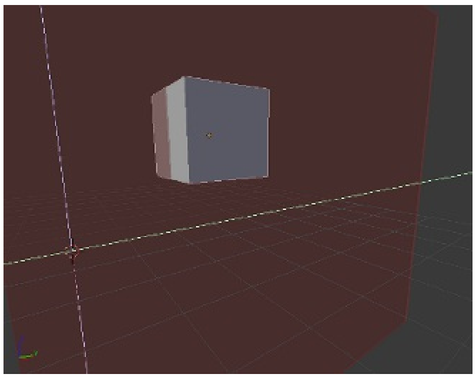
Axis-Constraint in action

MMB
: Constrain the move to the X, Y or Z axis automatically, according to the position of the mouse pointer in the 3D View. After pressing the G key, if the MMB is pressed, a visual option to constrain the translation will be available, showing the three axis in the 3D View space. The axis of choice to confirm the operation, will depend on the axis about which the MMB is released.

At any point during th eoperation, the chosen axis can be changed by hitting X, Y, Z on the keyboard.

RMB or Esc
: Cancel the move, and return the object or data to its original location.

Alt + g
: Clears all the previously done transformation on the object.Works only in Object Mode.


Shift+X in action

⇧ Shift and XYZ
: Complementary axis transformation constraint. With this option, we can isolate the transformation to axis complementary to the choosen axis. When a specific axis is choosen, the translation will occur in all axes other than the chooosen one. This can be seen in the example image

Controling Grab/Move Precision

In addition to the Axis constraint options listed above, Blender offer some options to limit the amount of the transformation in small or predefined steps.

⇧ Shift
: Slow transformation option. While still in the grab mode i.e.,after G pressed, if the ⇧ Shift key is pressed, the rate of transformation is reduced giving you precise translation.

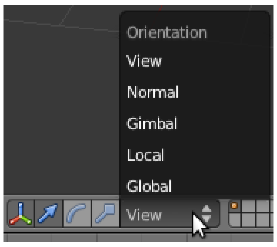
Ctrl
: Snap while grabbing the object based on the snapping constraint which has been already set. For this option you may not necessarily enable the snap option. It will work from the snap disabled mode itself.

Ctrl+⇧ Shift
: Intuitively this is the combination of the Ctrl and the ⇧ Shift operations individually. This option will move the object with high precission along with the snapping constraint.

X/y/z + <decimal number>
: This option will limit the transformation to the specified axis and the decimal number specified will be the magnitude of the translation along that axis. This decimal number which is being entered will be displayed at the bottom left corner of the 3D view window. Hitting backspace during the number entry will remove the numerical specification option but the object will be in the same axis.The number can be retyped to specify the translation.At any point of time, axis can be changed by hitting x/y/z key. You can also use this to move to a specific location or increase distance for the object location.

Orientations

There are 5 orientations for all tranformations.



Orientation choice menu

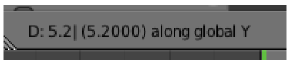
- Global(default)
- Local
- Normal
- Gimbal
- View

Read more about transform orientations Here

Each mode is a co-ordinate system in which the transformations can be carried out. These orientations can be chosen from the pop up menu just by the side of transformation manipulator choice widget group.

G key followed by Xx or Yy or Zz will directly allow you to translate the objects in local axis. Of course this can also be followed by numerical specification of the displacement of entity.

Similar to above operation, G key followed by ⇧ Shift and Xx or Yy or Zz will directly allow you to translate the objects in local axis complementary to the one specified.


D: 5.2| (5.2000) along global Y

Numerical Entry Display'

Other Editor Windows

For the other Editor Windows, like UV/Image Editor Tools, Sequence Editor, Dopesheet, and Graph Editor, the Grab/Move Operations are used to move Objects or elements based in their context, but, differently from the 3D View, you will see only two axis, **X** and **Y** normally, and altough we are explainning the Grab/Move in the **3D Interaction** section, those Objects and elements are shown in a 3D Interface. Blender will simply constrain the movement of a third possible axis. Most of the shortcuts used in the 3D View, are also used when interacting with those Editor Windows. This is also true for all of the other transformations, like rotate and scale.

Python Scripting

You can also use Python Scripting in Blender to Grab/Move Objects or elements to a specific location, either using the Python interactive console, or running a Python script in the Text Editor Window.

Getting the location vector for current object
    bpy.context.scene.objects.active.location

Returns you the location vector for the active object in the scene.One can assign a different value to the location vector to change the position of the object.

Operator for translating active object and its syntax
    bpy.ops.transform.translate(value=(<DX>, <DY>, <DZ>), constraint_axis=(<bool>, <bool>,<bool>), constraint_orientation='<ORIENTATION NAME>', mirror=<bool>, proportional='<ENABLE?DISABLE>', proportional_edit_falloff='<FALLOFF TYPE>', proportional_size=<INT>, snap=<bool>, snap_target='<SNAP TARGET>', snap_point=<x,y,z>, snap_align=<bool>, snap_normal=<x,y,z>, texture_space=<bool>, release_confirm=<bool>)

Hints

- Moving object in Object mode is clearly different from moving the object by selecting all its vertices/edges/faces in Edit mode. Doing this can lead to disturbed Center of Transformation for the given object.
- If G+x/y/z is used in non global orientations, it won't confine the translation to x axis in that orientation but to the global X axis orientation only

Rotate

Mode: Object and Edit modes

Hotkey: R

Menu: Object/Mesh/Curve/Surface » Transform » Rotate

Description

Rotation is also known as a spin, twist, orbit, pivot, revolve, or roll and involves changing the orientation of elements (vertices, edge, face, Object etc) around one or more axes or the element's Pivot Point. There are multiple ways to rotate an element which include:

1. The keyboard shortcut (R)
2. The 3D manipulator widget
3. The Properties menu (N)

Basic rotation usage and common options are described below. For additional information, you may wish to read the Transform Control and Orientation pages which provide more information about options such as Precision, Axis Locking, Numeric Input, Snapping and the different types of Pivot Point.

Read more about Transform Control »
Read more about Transform Orientations »

Usage

Rotation using the keyboard shortcut

1. Use RMB to select the elements you want to rotate.
2. Tap R once to enter rotation mode.
3. Rotate the elements by moving the mouse. The closer the mouse is to the elements's center, the higher the rotation influence.
4. LMB click to accept changes.

The amount of rotation will be displayed in the bottom left hand corner of the 3D window.

Rot: 26.36

Rotation values

Constraining the rotation axis (axis locking)

Rotation can be constrained to a particular axis or axes through the use of Axis Locking. To constrain rotation, the following shortcuts can be used:

- R, X: Rotate only along the **X Axis**
- R, Y: Rotate only along the **Y Axis**
- R, Z: Rotate only along the **Z Axis**

Axis locking can also be enabled by pressing the MMB after enabling rotation and moving the mouse in the desired direction e.g.

- R, move the mouse along the X axis, MMB: Rotate only along the **X Axis**

Read more about Axis Locking »

Fine Tuning The Rotation

Precise control can be had over rotation through the use of the ⇧ Shift and Ctrl keys to limit rotation to discrete amounts. You can also enter a numerical value in degrees to specify the amount of rotation after after initiating a rotation transformation.

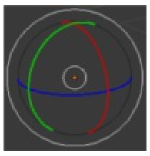
- Hold Ctrl down while performing a rotation to rotate the selected element in 5 degree increments.
- Hold ⇧ Shift down while performing a rotation to rotate the selected element in 0.01 degree increments.
- Hold ⇧ ShiftCtrl down while performing a rotation to rotate the selected element in 1 degree increments.
- Press R, type in a number and press ↵ Enter to confirm.
- Press R,R to enable Trackball rotation.

Orientation dependant rotations

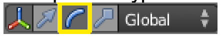
By default, all rotations happen around a Global Orientation. You can change the rotation orientation by pressing the axis key twice. For example, pressing R, X, X will by default set rotation to occur around the local orientation.

Read more about Precision Control »
Read more about Numerical Transformations »
Read more about Transform Orientations »

Rotation with the 3D Transform Manipulator


Rotation Transform Manipulator

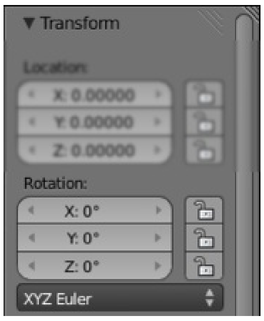
In the 3D View header, ensure that the Transform Manipulator is enabled (the red, green, and blue triad is selected). Set the manipulator type to rotation (the highlighted arc icon shown below).



1. Select your element with RMB.
2. Use LMB and drag any of the three colored axes on the rotation manipulator to rotate your object along that axis. You can also use ⇧ Shift, Ctrl or numeric input with the 3D manipulator widget for further control.
3. Your changes will be applied when you release LMB or press Space or ↵ Enter. Your changes will be cancelled if you press RMB or Esc.

Read more about the 3D Transform Manipulator »

Rotation with the Properties Panel



Rotation transform properties panel.

Rotation values can also be specified in the Properties panel (N) by altering the degree value in the rotation slider of the Transform panel. Rotation along particular axes can be enabled or disabled by toggling the padlock icon. The rotation mode (Euler, Axis Angle, Quaternion) can also be set in this panel from the drop down box.

Read more about Panels »
Read more about rotation modes »
Additional detail about rotation modes »

Scale

Mode: Object and Edit modes

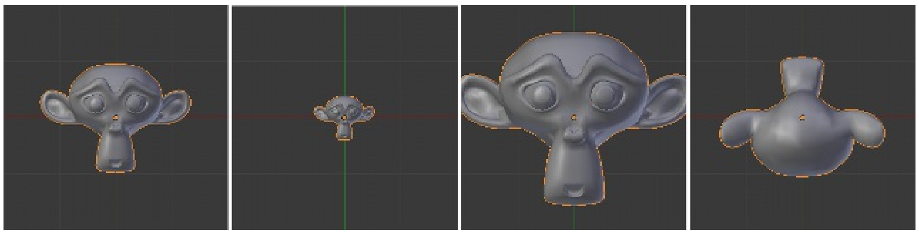
Hotkey: S

Menu: Object/Mesh/Curve/Surface » Transform » Scale

Description

Pressing S will enter the Scale transformation mode where the selected element is scaled inward or outward according to the mouse pointer's location. The element's scale will increase as the mouse pointer is moved away from the Pivot Point and decrease as the pointer is moved towards it. If the mouse pointer crosses from the original side of the Pivot Point to the opposite side, the scale will continue in the negative direction and flip the element.

Read more about Pivot Points »


Basic scale usage. From left to right, the panels show: the original Object, a scaled down Object, a scaled up Object and a scale-flipped Object.

There are multiple ways to scale an element which include:

1. The keyboard shortcut (S)
2. The 3D manipulator widget
3. The Properties menu (N)

Basic scale usage and common options are described below. For additional information, you may wish to read the Transform Control and Orientation pages which provide more information about options such as Precision, Axis Locking, Numeric Input, Snapping and the different types of Pivot Point.

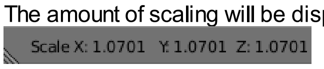
Read more about Transform Control »
Read more about Transform Orientations »

Usage

Scaling using the keyboard shortcut

1. Use RMB to select the elements you want to scale.
2. Tap S once to enter scale mode.
3. Scale the elements by moving the mouse.
4. LMB click to accept changes.

The amount of scaling will be displayed in the bottom left hand corner of the 3D window.


Scale X: 1.0701 Y: 1.0701 Z: 1.0701

Scale values

Constraining the scaling axis (axis locking)

Scaling can be constrained to a particular axis or axes through the use of Axis Locking. To constrain scaling, the following shortcuts can be used:

- S, X: Scale only along the **X Axis**
- S, Y: Scale only along the **Y Axis**
- S, Z: Scale only along the **Z Axis**

Axis locking can also be enabled by pressing the MMB after enabling scaling and moving the mouse in the desired direction e.g.

- S, move the mouse along the X axis, MMB: Scale only along the **X Axis**

Read more about Axis Locking »

Fine Tuning The Scaling

Precise control can be had over scaling through the use of the ⇧ Shift and Ctrl keys to limit scaling to discrete amounts. You can also enter a numerical value in Blender Units (BU) to specify the amount of scaling after after initiating a scale transformation.

- Hold Ctrl down while scaling to scale the selected element in degree 0.1 BU increments.
- Hold ⇧ Shift down while scaling to scale the selected element in very fine increments.

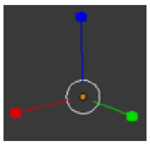
- Hold ⇧ ShiftCtrl down while scaling to scale the selected element in 0.01 BU increments.
- Press S, type in a number and press ↵ Enter to confirm.

Orientation dependent scaling

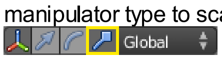
By default, all scaling happens around a Global Orientation. You can change the scaling orientation by pressing the axis key twice. For example, pressing S, X, X will by default set scaling to occur around the local orientation.

Read more about Precision Control »
Read more about Numerical Transformations »
Read more about Transform Orientations »

Scaling with the 3D Transform Manipulator


Scaling Transform Manipulator

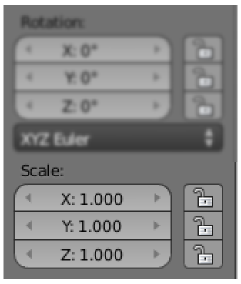
In the 3D View header, ensure that the Transform Manipulator is enabled (the red, green, and blue triad is selected). Set the manipulator type to scale (the highlighted square icon shown below).



1. Select your element with RMB.
2. Use LMB and drag any of the three colored axes on the scaling manipulator to scale your object along that axis. You can also use ⇧ Shift, Ctrl or numeric input with the 3D manipulator widget for further control.
3. Your changes will be applied when you release LMB or press Space or ↵ Enter. Your changes will be cancelled if you press RMB or Esc.

Read more about the 3D Transform Manipulator »

Scaling with the Properties Panel



Scale transform properties panel.

Scale values can also be specified in the Properties panel (N) by altering the amount value in the scaling slider of the Transform panel. Scaling along particular axes can be enabled or disabled by toggling the padlock icon.

Read more about Panels »
Read more about scaling modes »

Page status (reviewing guidelines)

Void page
Proposed fixes: none

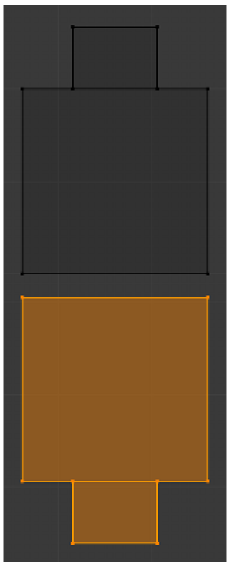
Mirror

Mode: Object and Edit modes

Hotkey: CtrlM

Menu: Object/Mesh » Mirror

Description


Mirroring a selection.

Mirroring an Object or Mesh selection will create a reversed version of the selection. The position of the mirrored version of the selection is determined by the Pivot Point. A common use of mirroring is to model half an object, duplicate it and then use the mirror transform to create a reversed version to complete the model. Note that mirrored duplicates can also be created with a Mirror modifier.

Read more about the Pivot Point »
Read more about the Mirror Modifier »

Usage

To mirror a selection along a particular global axis press:

CtrlM, followed by X, Y or Z.

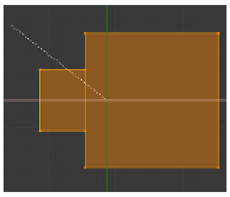
The image Mirroring a selection shows the results of this action after a mesh element has been duplicated.

In Mesh mode, you can mirror the selection on the currently selected Transform Orientation by pressing the appropriate axis key a second time. For example, if the Transform Orientation is set to Normal, pressing:

CtrlM, followed by X and then X again

will mirror the selection along the X-axis of the Normal Orientation.

Read more about Transform Orientations »


Interactive mirror.

You can alternatively hold the MMB to interactively mirror the object by moving the mouse in the direction of the mirror axis.

To Sphere

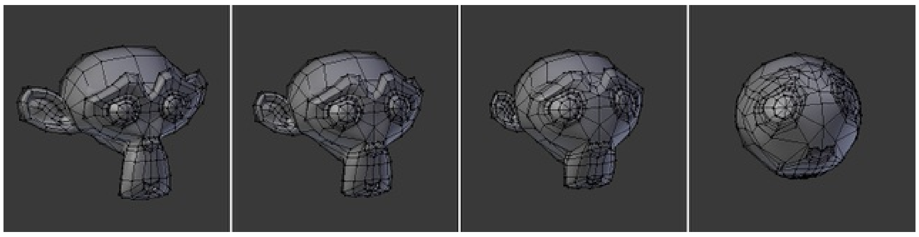
Mode: Edit mode

Hotkey: ⇧ ShiftAltS

Menu: Mesh » Transform » To Sphere

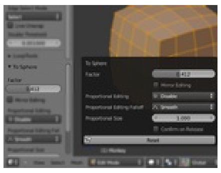
Description

The To Sphere transformation will give the selection spherical qualities. The *Suzanne with increasing sphericity* image below shows the results of applying the To Sphere transformation to the Suzanne mesh.



Suzanne with inceasing sphericity. The sequence above shows a Suzanne mesh with a 0, 0.25 (25%), 0.5 (50%) and 1 (100%) To Sphere transform applied.

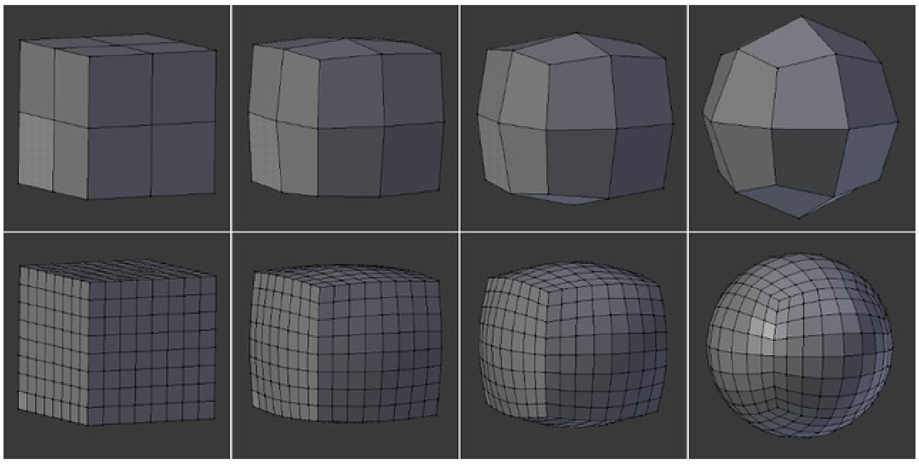
Usage



To Sphere Factor.

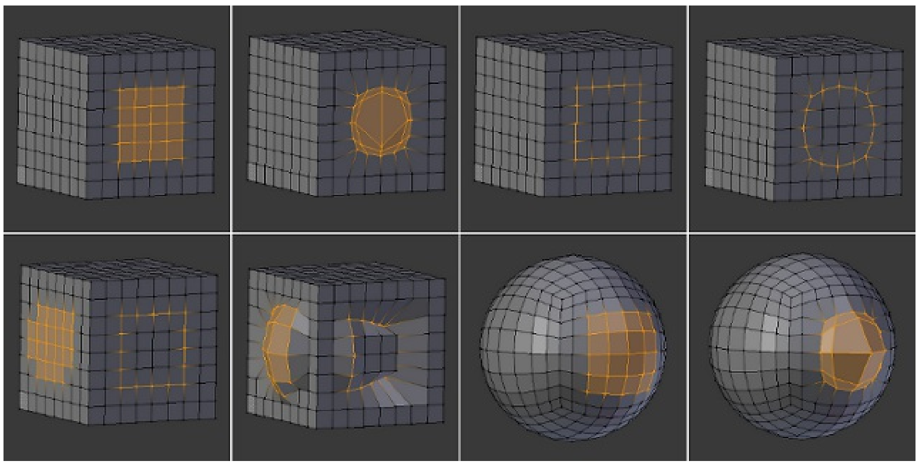
Select the elements you want to operate on and activate the To Sphere transform function. The To Sphere option can be invoked from the Mesh » Transform » To Sphere menu option or by pressing ⇧ ShiftAltS. The amount of sphericity given to the selection can be determined interactively by moving the mouse or by typing a number between 0 and 1. Pressing ↵ Enter will confirm the transformation. The confirmed transformation can be further edited by pressing F6 or by going into the Toolshelf (T) and altering the Factor slider provided that no other actions take place between the To Sphere transform confirmation and accessing the slider.

Note that the result of the To Sphere transform is also dependant on the number of selected mesh elements (vertices, faces etc). As can be seen in the below image, the result will be smoother and more spherical when there are more mesh elements available to work with.



To Sphere applied to cubes with different subdivision levels. In this image sequence, To Sphere was applied to the entire cube at levels of 0, 0.25 (25%), 0.5 (50%) and 1 (100%) respectively.

The To Sphere transform will generate different results depending on the number and arrangement of elements that were selected (as shown by the below image).


To Sphere applied to different selections.

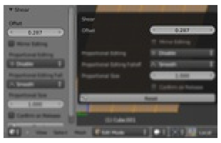
Shear

Mode: Object and Edit modes

Hotkey: ⇧ ShiftCtrlAltS

Menu: Object/Mesh/Curve/Surface » Transform » Shear

Description


Shear Offset Factor.

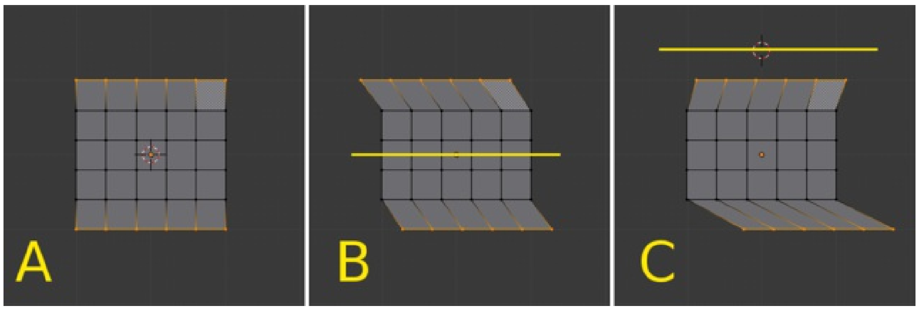
Shearing is a form of movement where parallel surfaces move past one another. During this transform, movement of the selected elements will occur along the horizontal axis of the current view. The axis location will be defined by the Pivot Point. Everything that is “above” this axis will move (Shear) in the same direction as your mouse pointer (but always parallel to the horizontal axis). Everything that is “below” the horizontal axis will move in the opposite direction.

Read more about Pivot Points »

Usage

Select the elements you want to operate on and activate the Shear transform function. The Shear option can be invoked from the Object/Mesh/Curve/Surface » Transform » Shear menu option or by pressing ⇧ ShiftCtrlAltS. The amount of movement given to the selection can be determined interactively by moving the mouse or by typing a number. Pressing ↵ Enter will confirm the transformation. The confirmed transformation can be further edited by pressing F6 or by going into the Toolshelf (T) and altering the Offset slider provided that no other actions take place between the Shear transform confirmation and accessing the slider.

Note that the result of the Shear transform is also dependant on the number and type of selected elements (Objects, vertices, faces etc). See below for the result of using Shear on a number of different elements.

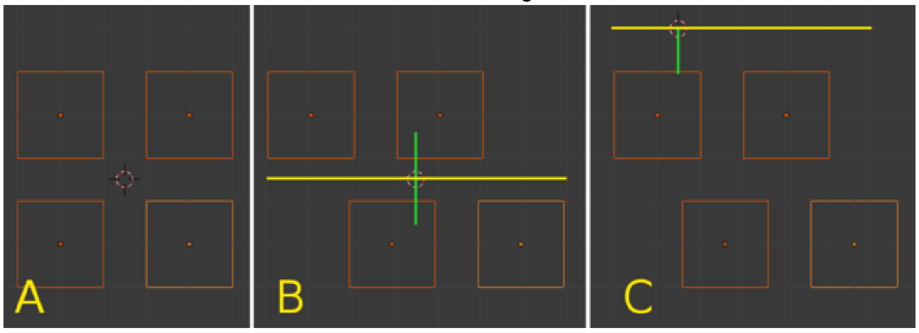


The effects of a Shear transform with different Pivot Points. See the text below for additional information.

The three frames of the image above show the effects of shearing on the selected vertices when the pivot point is altered. In frame B, the Pivot Point is set to Median Point (indicated by the yellow line) and the mouse was moved to the left during the transform. In frame C, the Pivot Point is set to the 3D cursor which is located above the mesh (indicated again by the yellow line). When the mouse is moved to the left during a Shear transform the selected vertices are moved to the right as they are below the horizontal axis.

Shear transform magnitude

The magnitude of the Shear transform applied to the selected elements is directly proportional to the distance from the horizontal axis. i.e. the further from the axis, the greater the movement.



The effects of a Shear transform on Objects with different Pivot Points. See the text below for additional information.

The three frames of the image above show the effects of shearing on the selected Objects when the Pivot Point is altered. In frame B, the Pivot Point is set to Median Point (indicated by the yellow line) and the mouse was moved to the left during the transform. In frame C, the Pivot Point is set to the 3D cursor which is located above the Objects (indicated again by the yellow line). When the mouse is moved to the left during a Shear transform all of the selected Objects are moved to the right as they are below the horizontal axis.

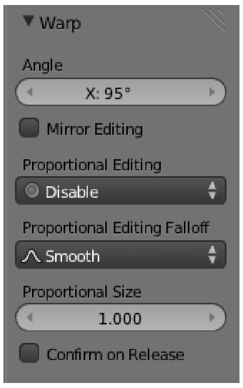
Again, note that the magnitude of the transform is proportional to the distance from the horizontal axis. In this case, the lower Objects move further than the upper ones.

Warp

Mode: Object and Edit modes

Hotkey: ⇧ ShiftW

Menu: Object/Mesh/Curve/Surface » Transform » Warp


▼ Warp

Angle
X: 95°

☐ Mirror Editing

Proportional Editing
Disable

Proportional Editing Falloff
Smooth

Proportional Size
1.000

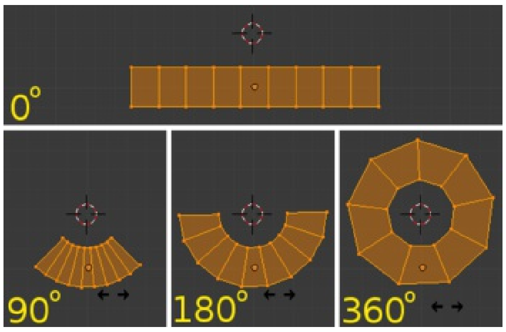
☐ Confirm on Release

warp tool options

In Edit mode, the Warp transformation takes selected elements and warps them around the 3D cursor by a certain angle. Note that this transformation is always dependent on the location of the 3D cursor. The Pivot Point is not taken into account. The results of the Warp transformation are also view dependent.

In Object mode, the Warp transformation takes the selected Objects and causes them to move in an orbit-like fashion around the 3D cursor. Similar to Edit mode, the Pivot Point is not taken into account and the results are view dependent.

Usage



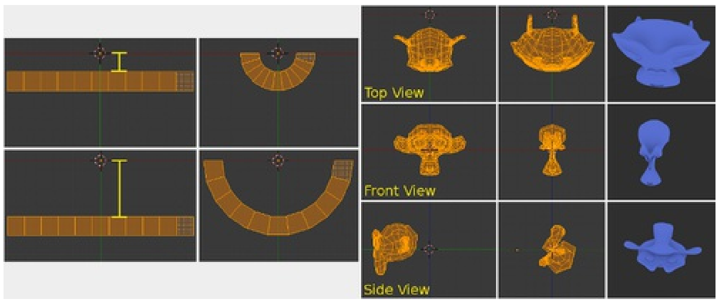
In this example, a plane is warped around the 3D cursor by the indicated number of degrees.

Select the elements you want to operate on and activate the Warp transform function. The Warp option can be invoked from the Object/Mesh/Curve/Surface » Transform » Warp menu option or by pressing ⇧ ShiftW. The amount of warping given to the selection can be determined interactively by moving the mouse or by typing a number. Pressing ↵ Enter will confirm the transformation. The confirmed transformation can be further edited by pressing F6 or by going into the Toolshelf (T) and altering the Angle slider provided that no other actions take place between the Warp transform confirmation and accessing the slider.

Cursor position and view

The location of the 3D cursor can be used to alter the results of the Warp transformation. As can be seen from the example in this section, the Warp radius is dependent on the distance of the cursor from the selected elements. The greater the distance, the greater the radius.

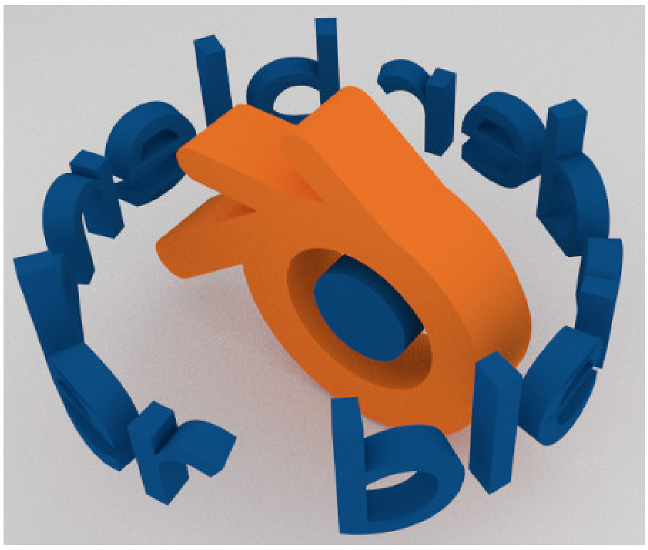
The result of the Warp transform is also influenced by your current view. The example in this section shows the results of a 180 degree Warp transform applied to the same Suzanne mesh when in different views. A 3D render is also provided for comparison.



The left side of this image shows how the Warp transform is influenced by the location of the cursor. The right hand side shows the influence of the current view.

Warping text
If you want to warp text, you will need to convert it from a Text Object to Mesh by pressing AltC and selecting the Mesh from Curve/Meta/Surf/Text option.

Example

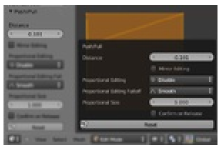

Text wrapped around logo. This was made by creating the Blender logo and text as separate Objects. The text was converted to a mesh and then warped around the Blender logo.

Push/Pull

Mode: Object and Edit modes

Menu: Object/Mesh » Transform » Push Pull

Description

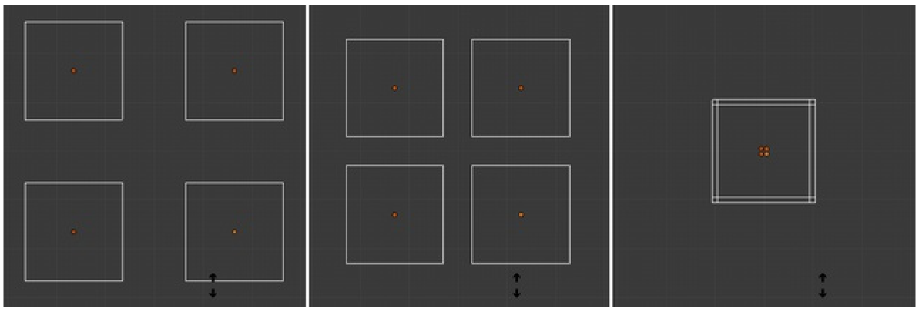

Push/Pull distance.

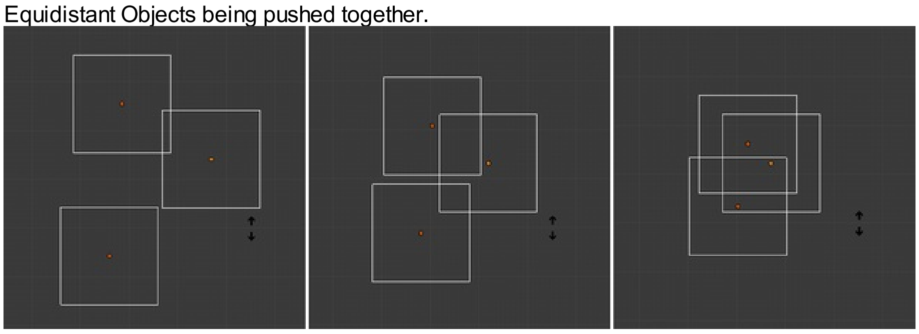
Push/Pull will move the selected elements (Objects, vertices, edges or faces) closer together (Push) or further apart (Pull). Specifically, each element is moved towards or away from the center by the same distance. This distance is controlled by moving the mouse up (Push) or down (Pull), numeric input or through slider control.

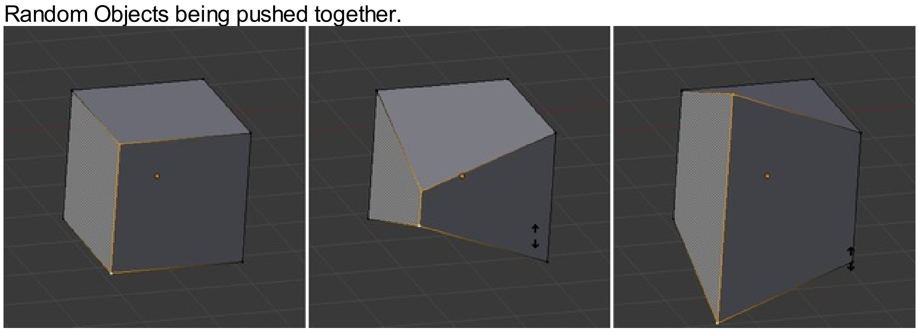
Usage

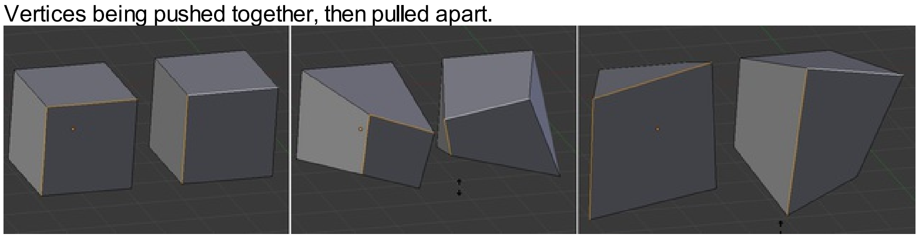
Select the elements you want to operate on and activate the Push/Pull transform function. The Push/Pull option can be invoked from the Object/Mesh » Transform » Push/Pull menu option or by pressing Space and using the search menu to search for Push or Pull. The amount of movement given to the selection can be determined interactively by moving the mouse or by typing a number. Pressing ↵ Enter will confirm the transformation. The confirmed transformation can be further edited by pressing F6 or by going into the Toolshelf (T) and altering the Distance slider provided that no other actions take place between the Push/Pull transform confirmation and accessing the slider.

Note that the result of the Push/Pull transform is also dependant on the number and type of selected elements (Objects, vertices, faces etc). See below for the result of using Push/Pull on a number of different elements.


Equidistant Objects being pushed together.


Random Objects being pushed together.

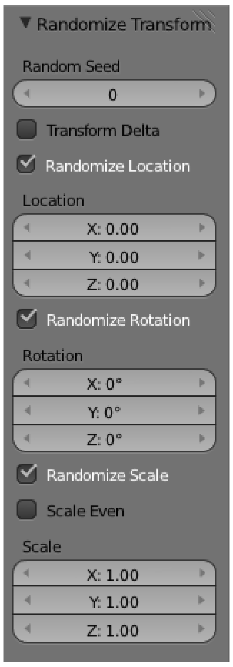

Vertices being pushed together, then pulled apart.


Edges on separate meshes being pushed together, then pulled apart.

Randomize Transform

Mode: Object mode

Menu: Object » Transform » Randomize Transform


Randomize Transform
Random Seed: 0
Transform Delta: ☐
Randomize Location: ✓
Location: X: 0.00, Y: 0.00, Z: 0.00
Randomize Rotation: ✓
Rotation: X: 0°, Y: 0°, Z: 0°
Randomize Scale: ✓
Scale Even: ☐
Scale: X: 1.00, Y: 1.00, Z: 1.00

Randomize transform options

The randomize transform tool allows you to apply random translate, rotate, and scale values to an object or multiple objects. When applied on multiple objects, each object gets its own seed value, and will get different transform results from the rest.

Options

Random Seed
: The random seed is an offset to the random transformation. A different seed will produce a new result.

Transform Delta
: Randomize Delta Transform values instead of regular transform. See Delta Transforms.

Randomize Location
: Randomize Location vales

Location
: The maximum distances the objects can move along each axis.

Randomize Rotation
: Randomize rotation values.

Rotation
: The maximum angle the objects can rotate on each axis

Randomize Scale
: Randomize scale values.

Scale Even
: Use the same scale for each axis.

Scale
: The maximum scale randomization over each axis.

Transform Control

Transform controls can be used to modify and control the effects of the available transformations.

The following pages detail the available control options:

Transformation Amount

- Precision of Transformations
- Numeric Transformations
- Transform Properties
- Reset Object Transforms
- Proportional Edit

Transformation Orientation

- Manipulators
- Transform Orientations
- Axis Locking

Transformation Center

- Pivot Point
- Active object
- Individual Centers
- 3D Cursor
- Median Point
- Bounding Box Center

Transformation Snapping

- Snapping Transformations
- Snap to Mesh

Precision

Mode: Object and Edit modes

Hotkey: Ctrl and/or ⇧ Shift

Description

Holding Ctrl or ⇧ Shift during a transformation operation (such as grab/move, rotate or scale) will allow you to perform the transformation in either fixed amounts, very small amounts or both. The magnitude of the transformation can be viewed in the 3D window header in the bottom left hand corner. Releasing Ctrl or ⇧ Shift during the transformation will cause the movement to revert back to its normal mode of operation.

Read more about Window Headers »

Usage

With hotkeys

Press G, R or S and then hold either Ctrl, ⇧ Shift or Ctrl⇧ Shift.

With the Transform Manipulator

Hold Ctrl, ⇧ Shift or Ctrl⇧ Shift and click on the appropriate manipulator handle. Then move the mouse in the desired direction. The reverse action will also work i.e. clicking the manipulator handle and then holding the shortcut key for precision control.

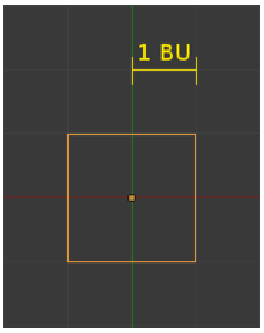
Read more about the Transform Manipulator »

Combining with other controls

All of the precision controls detailed on the page can be combined with the Axis Locking controls and used with the different Pivot Points.

Holding CTRL

Grab/move transformations



1 Blender Unit - shown at the default zoom level.

For grab/move operations at the default zoom level, holding Ctrl will cause your selection to move by increments of 1 Blender Unit (1 BU) (i.e. between the two light grey lines). Zooming in enough to see the next set of grey lines will now cause Ctrl movements to occur by 1/10 of a BU. Zooming in further until the next set of grey lines becomes visible will cause movement to happen by 1/100 of a BU and so on until the zoom limit is reached. Zooming out will have the opposite effect and cause movement to happen by increments of 10, 100 etc BU.

Read more about Zooming »

Rotation transformations

Holding Ctrl will cause rotations of 5 degrees.

Scale transformations

Holding Ctrl will cause size changes in increments of 0.1 BU.

Snapping modes
Note that if you have a Snap Element option enabled, holding Ctrl will cause the selection to snap to the nearest element.

Read more about Snapping »

Holding SHIFT

Holding ⇧ Shift during transformations allows for very fine control that does not rely on fixed increments. Rather, large movements of the mouse across the screen only result in small transformations of the selection.

Holding CTRL and SHIFT

Grab/move transformations

For grab/move operations at the default zoom level, holding Ctrl⇧ Shift will cause your selection to move by increments of 1/10 Blender Units. Holding Ctrl⇧ Shift at any zoom level will cause the transformation increments to always be 1/10 of the increment if you were only holding Ctrl.

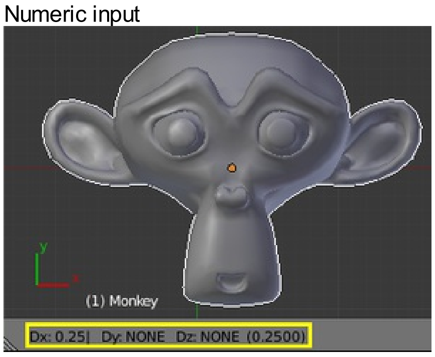
Rotation transformations

Holding Ctrl⇧ Shift will cause rotations of 1 degree.

Scale transformations

Holding Ctrl⇧ Shift will cause size changes in 0.01 BU increments.

Numeric input



Dx: 0.25| Dy: NONE Dz: NONE (0.2500)

Numeric input in the 3D window header

Using the mouse for transformations is convenient, but if you require more precise control, you can also enter numeric values. After pressing G, R or S, type a number to indicate the magnitude of the transformation.

You can see the numbers you enter in the bottom left hand corner of the 3D window header. Negative numbers and decimals can be entered by pressing the minus (-) and period (.) keys respectively.

Translation

To move Objects, vertices, faces or edges select the element, press G and then type a number. By default and with no other key presses, movement will occur along the X-axis. To confirm the movement, press ↵ Enter or LMB. To cancel the movement, press Esc or RMB. If you mistype the value, press ← Backspace to cancel the current entry and retype a new value.

To enter numeric values for multiple axes, use the ⇆ Tab key after entering a value for the axis. e.g. To move an Object, one (1) Blender unit on all three axes press: G, 1, ⇆ Tab, 1, ⇆ Tab, 1. This will move the element one unit along the X-axis, followed by the Y-axis and then the Z-axis.

You can also combine numeric input with axis locking to limit movement to a particular axis. To do so, press G followed by X, Y or Z to indicate the axis. Then type in the transform amount using 0-9 followed by ↵ Enter to confirm. Pressing X, Y or Z will initially constrain movement to the Global axis. Pressing X, Y or Z again will constrain movement to the orientation set in the Transform Orientation setting of the 3D window header.

Read more about Transform Orientations »

Read more about Axis Locking »

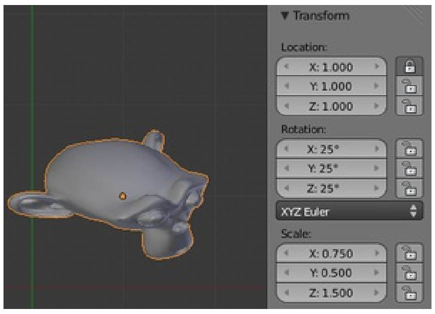
Rotation

To specify a value for clockwise rotation, press R, 0-9, then ↵ Enter to confirm. To specify counter-clockwise rotation press R, -, 0-9, then ↵ Enter to confirm. Note that 270 degrees of clockwise rotation is equivalent to -90 degrees of counter-clockwise rotation.

Scaling

Objects, faces and edges can be scaled by pressing S, 0-9, then ↵ Enter to confirm., Scaling transformations can also be constrained to an axis by pressing X, Y or Z after pressing S. Essentially, scaling with numeric values works in almost identical fashion to translation. The primary difference is that by default, scaling applies equally to all three axes. e.g. pressing S, 0.5, ↵ Enter will scale an Object by 0.5 on all three axes.

Numeric input via the Properties shelf



Location:
X: 1.000
Y: 1.000
Z: 1.000
Rotation:
X: 25°
Y: 25°
Z: 25°
XYZ Euler
Scale:
X: 0.750
Y: 0.500
Z: 1.500

Transformations can also be entered through the Transform panel on the Properties shelf.

It is also possible to enter numeric values for each transformation using the Transform panel found on the Properties shelf (N). The Transform panel can also be used to prevent transformations along particular axes by clicking the lock icon.

Transform Properties

Each object stores its position, orientation, and scale values. These may need to be manipulated numerically, reset, or applied.

Transform Properties Panel

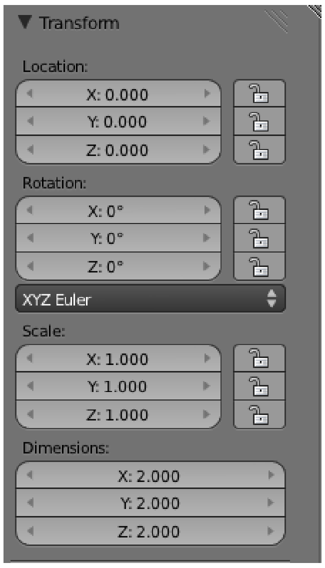
Mode: Edit and Object modes

Hotkey: N

Menu: Object » Transform Properties

The Transform Properties section in the View Properties panel allows you to view and manually/numerically control the position, rotation, and other properties of an object, in Object mode. In Edit mode, it mainly allows you to enter precise coordinates for a vertex, or median position for a group of vertices (including an edge/face). As each type of object has a different set of options in its Transform Properties panel in Edit mode, see their respective descriptions in the Modeling chapter.

Options in Object mode


Transform

Location:
X: 0.000
Y: 0.000
Z: 0.000

Rotation:
X: 0°
Y: 0°
Z: 0°
XYZ Euler

Scale:
X: 1.000
Y: 1.000
Z: 1.000

Dimensions:
X: 2.000
Y: 2.000
Z: 2.000

Transform Properties panel in Object mode.

Location X, Location Y, Location Z
: The object's center location in global coordinates.

Rotation X, Rotation Y, Rotation Z
: The object's orientation, relative to the global axes and its own center.

Scale X, Scale Y, Scale Z
: The object's scale, relative to its center, in local coordinates (i.e. the Scale X value represents the scale along the local X-axis). Each object (cube, sphere, etc.), when created, has a scale of one blender unit in each local direction. To make the object bigger or smaller, you scale it in the desired dimension.

Dimensions X, Dimensions Y, Dimensions Z
: The object's basic dimensions (in blender units) from one outside edge to another, as if measured with a ruler. For multi-faceted surfaces, these fields give the dimensions of the bounding box (aligned with the local axes – think of a cardboard box just big enough to hold the object).

Use this panel to either edit or display the object's transform properties such as position, rotation and/or scaling. These fields change the object's center and then affects the aspect of all of its *vertices* and faces.

Ipo Note
The values of the location, rotation, and scale can also be affected by an Ipo keyframe, so if there are Ipo keys associated with the object, be sure to reset them after making changes in this panel, or your changes will be lost when the frame is changed (Ipo keys override manually-set properties).

Some fields have extra functionality or features, such as scroll regions. When attempting to edit these types of fields it is easier to use {⇧ Shift LMB instead of just LMB. After you have edited a field click outside of the field's edit area or press ↵ Enter to confirm the changes. Changes will be reflected in the display window immediately. To cancel, hit Esc. For further descriptions of the other features of an edit field see The Interface section.

Transform Properties Locking

The locking feature of the Location, Rotation and Scale fields allows you to control a transform property solely from the properties panel. Once a lock has been activated any other methods used for transformation are blocked. For example, if you locked the

Location X field then you can’t use the mouse to translate the object along the global X axis. However, you can still translate it using the Location X edit field. Consider the locking feature as a rigid constraint only changeable from the panel.

To lock a field, click the padlock icon next to the field. The field is unlocked if the icon appears as (🔓), and it is locked if the icon appears as (🔒).

Clear Object transformations

Mode: Object mode

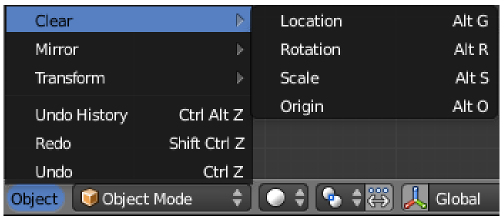
Hotkey: AltG, AltS, AltR, AltO

Menu: Object » Clear » Clear Location/Clear Scale/Clear Rotation/Clear Origin

Description

Clearing transforms simply resets the transform values. The objects location and rotation values return to 0, and the scale returns to 1.

Clear Options



Clear Transformation menu

Clear Location AltG
: Clear (reset) the location of the selection. This will move the selection back to the coordinates 0,0,0.

Clear Scale AltS
: Clear (reset) the scale of the selection. This will resize the selection back to the size it was when created.

Clear Rotation AltR
: Clear (reset) the rotation of the selection. This will set the rotation of the selection to 0 degrees in each plane.

Clear Origin AltO
: Clear (reset) the origin of the Child objects. This will cause Child objects to move to the coordinates of the parent.

Apply Object transformations

Mode: Object mode

Hotkey: CtrlA

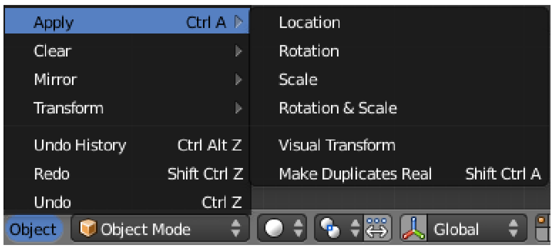
Menu: Object » Apply

Applying transform values essentially resets the values of object's position, rotation, or scale, but does not actually do anything to the object. The center point is moved to the origin and the transform values are set to zero. In terms of scale, the scale values return to 1.

To apply a transform select the Apply sub-menu from the Object menu or use the shortcut CtrlA and select the appropriate transform to apply

Make Duplicates Real unlinks linked duplicates so each duplicate now has its own datablock.

Apply Options



Apply Transformation menu

Apply Location CtrlA
: Apply (set) the location of the selection. This will make Blender consider the current location to be equivalent to 0 in each plane i.e. the selection will not move, the current location will be considered to be the "default location". The Object Center will be set to actual 0,0,0 (where the coloured axis lines intersect in each view).

Apply Rotation CtrlA
: Apply (set) the rotation of the selection. This will make Blender consider the current rotation to be equivalent to 0 degrees in each plane i.e. the selection will not rotated, the current rotation will be considered to be the "default rotation".

Apply Scale CtrlA
: Apply (set) the scale of the selection. This will make Blender consider the current scale to be equivalent to 0 in each plane i.e. the selection will not scaled, the current scale will be considered to be the "default scale".

Apply Rotation and Scale CtrlA

Apply (set) the rotation and scale of the selection. Do the above two applications simultaneously.

Apply Visual Transform CtrlA
: Apply (set) the result of a constraint and apply this back to the Object's location, rotation and scale. See the following post for more detailed discussion: Apply visual transform.

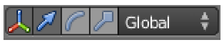
Make Duplicate Real ⇧ ShiftCtrlA
: Make any duplicates attached to this Object real so that they can be edited.

Manipulators

Mode: Object and Edit modes

Hotkey: CtrlSpace

In combination with axis locking, the normal Transform commands (G for Grab, R for Rotation, S for Scale), can be used to manipulate objects along any axis. However, there may be times when these options are not adequate. For example, when you want to translate a single face on a randomly rotated object in a direction perpendicular to the face's normal. In instances like this, Transform Manipulators may be useful.



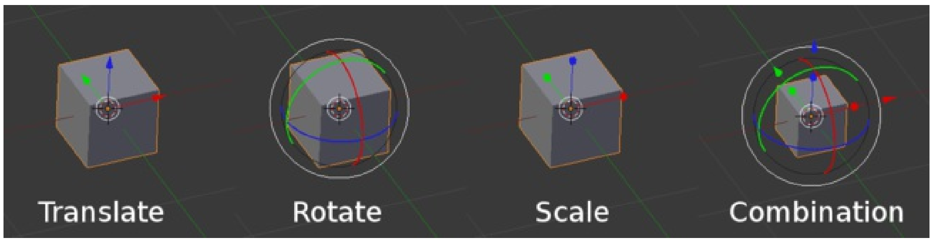
Manipulator options in the Window Header.

Transform manipulators provide a visual representation of the transform options and allow movement, rotation and scaling along any axis, mode and orientation of the 3D view. The manipulator can be enabled by clicking on the axis icon from the manipulator options portion of the window header or via the shortcut key CtrlSpace.

There is a separate manipulator for each Transform Command. Each manipulator can be used separately or in combination with the others. Clicking with ⇧ Shift LMB on multiple manipulator icons (arrow, arc, box) will combine manipulator options.

Manipulators can be accessed in the header of the 3D View window:

- Axis: Enable/disable the manipulators.
- Arrow: Translation.
- Arc: Rotation.
- Box: Scale.
- Transform Orientation menu: choice of the transformation orientation.



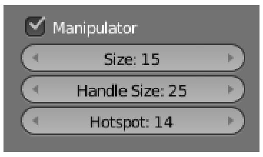
Manipulator Options

Manipulator controls

- Holding down Ctrl constrains the action to set increments. Holding down ⇧ Shift **after** you LMB the manipulator handle will constrain the action to smaller increments.
- Holding down ⇧ Shift **before** you LMB click on one of the handles will cause the manipulator action to be performed relative to the other two axes (you can let go of ⇧ Shift once you have clicked). For example, if you ⇧ Shift then LMB the Z axis handle of the translate manipulator, movement will occur in the X and Y planes.
- When in rotate mode, LMB on the white circle (largest circle around the rotation manipulator) will be equivalent to pressing R.
- When in rotate mode, LMB on the grey circle (small inner circle at the center of the rotation manipulator) will be equivalent to pressing R twice. This will start trackball rotation.

Read more about constraining transformations »
Read more about axis locking »
Read more about trackball rotation »

Manipulator Preferences


Manipulator

Size: 15

Handle Size: 25

Hotspot: 14

Manipulator preferences.

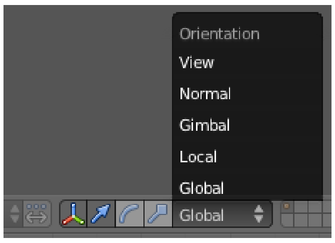
The settings of the manipulator (e.g. its size) can be found in the Interface section of the User Preferences window.

- Size: Diameter of the manipulator.
- Handle Size: Size of manipulator handles, as a percentage of the manipulator radius (Size/2).
- Hotspot: Hotspot size (in pixels) for clicking the manipulator handles.

Choosing the Transform Orientation

Mode: Object and Edit modes

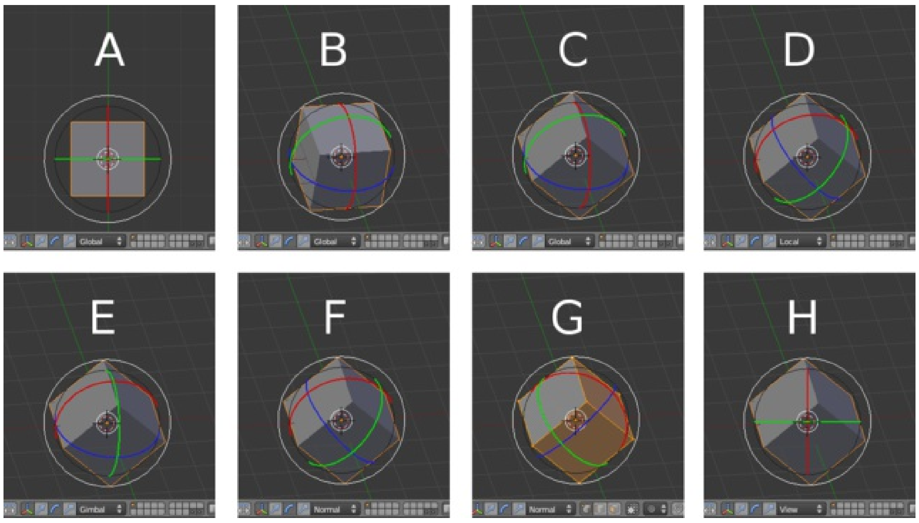
Hotkey: AltSpace



Transform Orientation options.

You can also change the orientation of the Transform Manipulator to global, local, gimbal, normal or view from the Transform options menu. The image below shows a cube with the rotation manipulator active in multiple transform orientations. Notice how the manipulator changes depending on the orientation selected (compare A with F).

Similarly, notice how when normal orientation (F and G) is selected the manipulator changes between Object mode and Edit mode. The normal orientation manipulator will also change depending on what is selected in Edit mode i.e. the orientation is based on the normal of the selection which will change depending on how many and which faces, edges or vertices are selected.



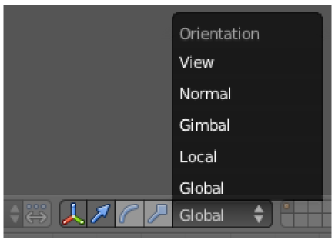
Transform manipulator orientation options.

- A: Standard cube in default top view with *global* orientation selected
- B: Standard cube with view rotated and *global* orientation selected
- C: Randomly rotated cube with view rotated and *global* orientation selected
- D: Randomly rotated cube with *local* orientation selected
- E: Randomly rotated cube with *gimbal* orientation selected
- F: Randomly rotated cube with *normal* orientation selected
- G: Randomly rotated cube, vertices selected with *normal* orientation selected
- H: Randomly rotated cube with *view*orientation selected

Transform Orientations

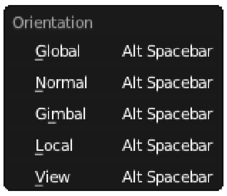
Mode: Object and Edit modes

Hotkey: AltSpace



Transform orientations selection menu.

Orientations affect the behavior of Transformations: Location, Rotation, and Scale. You will see an effect on the 3D Manipulator (the widget in the center of the selection), as well as on transformation constraints (like axis locking). This means that, when you hit GX, it will constrain to the global x-axis, but if you hit GXX it will constrain to your Transform Orientations x-axis.

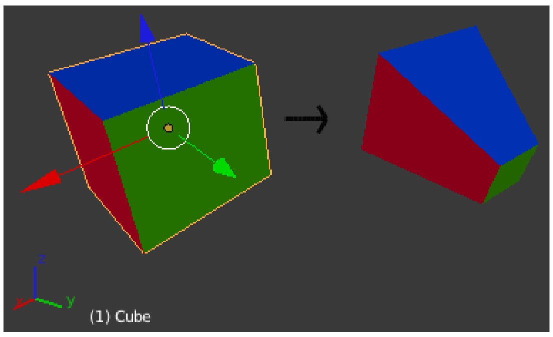


Alt+Space Menu.

The Orientations options can be set on the 3D View's header (or "footer", since it is at the bottom of the view by default), or with AltSpace or through the Orientation menu in a 3D view header.

In addition to the four built-in options, you can define a Custom Orientation.

Our Demo Cube

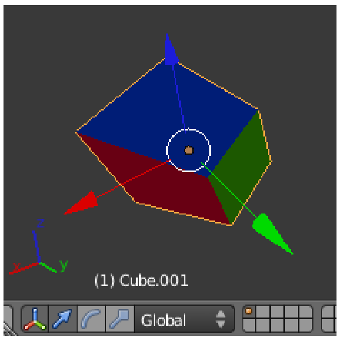


To demonstrate the various behaviors, we add some colors to the default cube, rotate it -15° along its local z- and x-axes, and we scale its "y" face down.

Please note two things:

- The "Mini-axis" in the lower-left corner, which represents the Global x/y/z orientation.
- The "Object Manipulator" widget emanating from the selection, which represents the current Transform Orientation.
  - If you click on one of the axes of the Manipulator with LMB, it will allow you to constrain movement to only this direction. An example of a keyboard equivalent is GZZ.
  - If you ⇧ Shift LMB click, it will lock the axis you clicked on and allow you to move in the plane of the two remaining axes. The keyboard analogue is G⇧ ShiftZ⇧ ShiftZ.

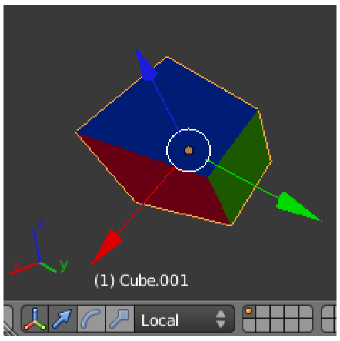
Orientations



Global.

Global

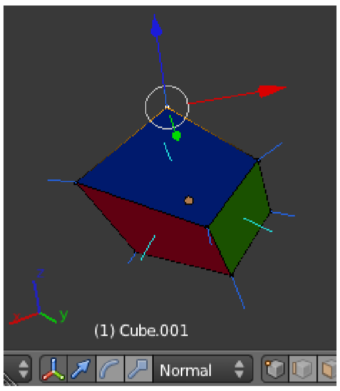
The manipulator matches the global axis. When using the Global orientation, the orientation's x,y,z matches world's x,y,z axis. When this mode is selected, the local coordinates of the object are subjected to the Global coordinates. This is good to place objects in the scene. To constrain an axis, hit G and the desired axis. To constrain to a local axis, hit the desired axis two times. The difference between Global and Local, is more noticeable when you have an object in which the origin is not located at the exact center of the object, and doesn't match the Global coordinates.



Local.

Local

The manipulator matches the object axis. Notice that, here, the Manipulator is at a slight tilt (it is most visible on the object's y-axis, the green arrow). This is due to our 15° rotation of the object. This demonstrates the difference between local coordinates and global coordinates. If we had rotated the object 90° along its x-axis, we would see that the object's "Up" is the world's "Forward" -- or the object's z-axis would now be the world's y-axis. This orientation has an effect on many parts of the interface, so it is important to understand the distinction.



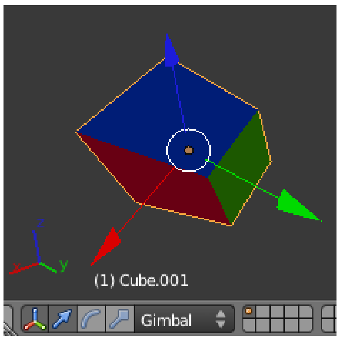
Normal.

Normal

The z-axis of the manipulator will match the normal vector of the selected object. In Object Mode, this is equivalent to Local Orientation, but in Edit Mode, it becomes more interesting.

As you see, the light blue lines indicate the faces' normals, and the darker blue lines indicate the vertex normals (these were turned on in the N Properties Panel under Mesh Display » Normals » Face and Vertex). Selecting any given face will cause our Manipulator's z-axis to align with that normal. The same goes for Vertex Select Mode. Edge Select is different--A selected Edge has the z-axis aligned with it (so you will have to look at the Manipulator widget to determine the direction of x and y). If you select several elements, it will orient towards the average of those normals.

A great example of how this is useful is in Vertex Select Mode: Pick a vertex and then do GZZ to tug it away from the mesh and shove it into the mesh. To make this even more useful, select a nearby vertex and hit ⇧ ShiftR to repeat the same movement---except along that second vertex's normal instead.



Gimbal.

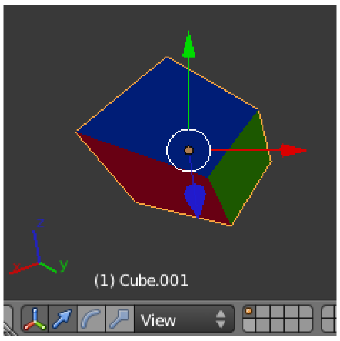
Gimbal

Gimbal's behavior highly depends on the Rotation Mode that you are in (accessible in the N Properties Panel in the 3D View, in top section, Transform).

XYZ Euler
: the default rotation mode, the object Manipulator's z-axis will always point to the global z-axis, where the other two will remain perpendicular to each other.
: In the other Euler rotation modes, the last axis applied will be the one for which the Manipulator stays fixed. So, for YZX Euler, the x-axis of the Manipulator will be the same as the global x-axis.

Axis Angle
: The x, y, and z coordinates define a point relative to the object origin through which an imaginary "skewer" passes. The w value is the rotation of this skewer. Here, the Manipulator's z-axis stays aligned with this skewer.

Quaternion
: Though Quaternion rotation is very different from the Euler and Axis Angle rotation modes, the Manipulator behaves the same as in Local mode.



View.

View

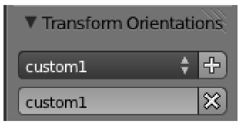
The manipulator will match the 3D view, Y → Up/Down, X → Left/Right, Z → Towards/Away from you.

This way you can constrain movement to one View axis with GXX.

Custom Orientations

Mode: Object and Edit modes

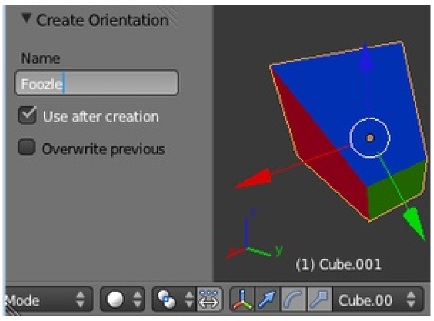
Hotkey: CtrlAltSpace



custom orientation

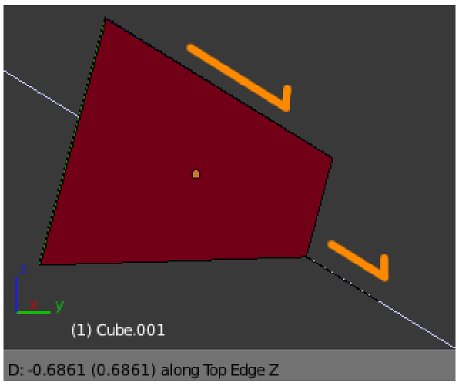
You can define custom transform orientations, using object or mesh elements. Custom transform orientations defined from objects use the local orientation of the object whereas those defined from selected mesh elements (vertices, edges, faces) use the normal orientation of the selection.

The Transform Orientations panel, found in the "N Properties Panel," can be used to manage transform orientations: selecting the active orientation, adding and deleting custom orientations.



Renaming a Custom Orientation

The default name for these orientations comes from whatever you have selected. If an edge, it will be titled, "Edge," if an object, it will take that object's name, etc. The Toolshelf (T in the 3D View) allows you to rename the custom orientation after you hit CtrlAltSpace.



D: -0.6861 (0.6861) along Top Edge Z

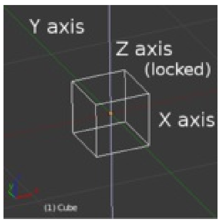
Figure 1.

The technique of creating custom orientations can become important in creating precise meshes. In Figure 1, to achieve this effect:

1. Select the object's sloping top edge
2. Create a Custom Orientation with CtrlAltSpace and rename it "Top Edge".
3. Select the objects's bottom, right edge.
4. Extrude with E.
5. Cancel the extrusion's default movement by hitting RMB or Escape.
6. Hit G to reinitiate movement.
7. Hit ZZ to constrain to the "Top Edge" orientation.

Axis Locking

Description



Axis locking

Transformations (translation/scale/rotation) in Object and Edit mode, as well as extrusion in Edit mode) can be locked to particular axis relative to the current transform orientation. By locking a transformation to a particular axis you are restricting transformations to a single dimension.

Usage

A locked axis will display in a brighter color than an unlocked axis. For example in the image to the right, the Z axis is drawn in light blue as movement is constrained to this axis. This example can be achieved in two ways:

- Press G to enable translation, press Z to constrain movement to the Z-axis.
- Press G to enable translation, move the mouse in the Z direction, then press MMB.

Axis locking types

Axis locking

Mode: Object and Edit modes (translate, rotate, scale, extrude)

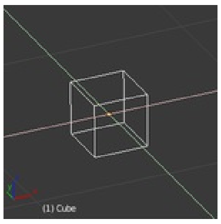
Hotkey: X, Y, Z or MMB after moving the mouse in the desired direction.

Axis locking limits the transformation to a single axis (or forbids transformations along two axes). An object, face, vertex or other selectable item will only be able to move, scale or rotate in a single dimension.

Plane locking

Mode: Object and Edit modes (translate, scale)

Hotkey: ⇧ ShiftX, ⇧ ShiftY, ⇧ ShiftZ or ⇧ Shift MMB after moving the mouse in the desired direction.

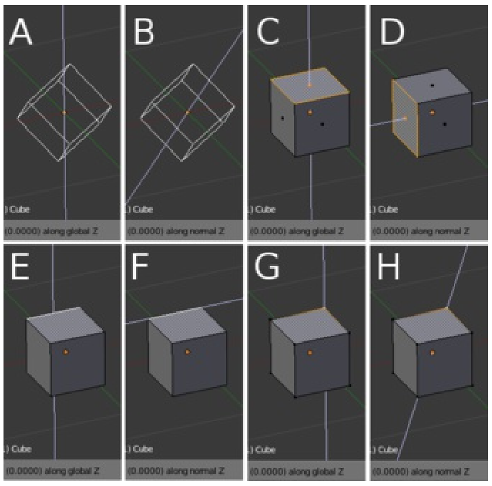


Plane locking

Plane locking locks the transformation to *two* axes (or forbids transformations along one axis), thus creating a plane in which the element can be moved or scaled freely. Plane locking only affects translation and scaling.

Note that for rotation, both axis and plane locking have the same effect because a rotation is always constrained around one axis. Trackball type rotations RR cannot be locked at all.

Axis locking modes



Axis locking modes

A single key press constrains movement to the corresponding Global axis. A second key press of the *same* key constrains movement to the current transform orientation selection (except if it is set to Global, in which case the Local orientation is used). Finally, a third key press of the same key removes constraints.

For example, if the current transform orientation is set to Normal, pressing G to start translation, followed by Z will lock translation in the Z direction relative to the Global orientation, pressing Z again will lock translation to the Z axis relative to the Normal orientation. Pressing Z again will remove all constraints. The current mode will be displayed in the left hand side of the 3D window header.

As can be seen in the *Axis locking modes* image, the direction of the transform also takes into account the selection. Sections A and B show Z axis locking in Global and Normal orientations respectively. C and D show the same situation with face selection, E and F with edge selection and G and H with vertex selection.

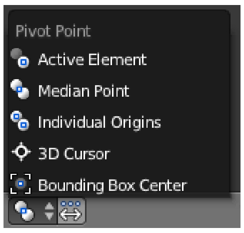
Note that using a locked axis does not prevent you from using the keyboard to enter numeric transformation values.

Pivot Point

Pivot Point selector

Mode: Object mode and Edit mode

Menu: Droplist in the header of the 3D view



Pivot Point modes.

The pivot point is the point in space around which all rotations, scalings and mirror transformations are centered. You can choose one of five Pivot Points from a drop-down list in the header of any 3D area, as seen here in (*Pivot Point modes*). The pages linked below describe each Pivot Point mode in more detail.

- Active element
- Median Point
- Individual Origins
- 3D Cursor
- Bounding Box Center

Note that even if the above examples use meshes, the same rules apply for other types (curves, surfaces…) as well.

Active Element as Pivot

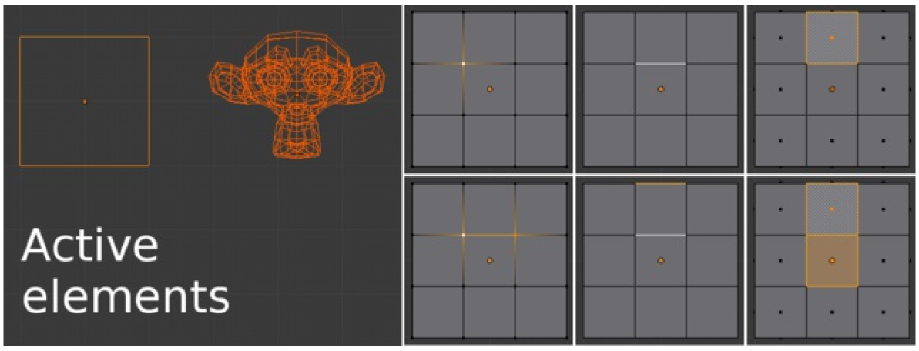
Mode: Object mode and Edit mode

Hotkey: Alt.

Menu: Select from the following icon in the 3D window header

The *active* element can be an Object, vertex, edge or a face. The active element is the last one to be selected and will be shown in a lighter orange color when in Object mode and white when in Edit mode. With Active element as Pivot set to active, all transformations will occur relative to the active element.

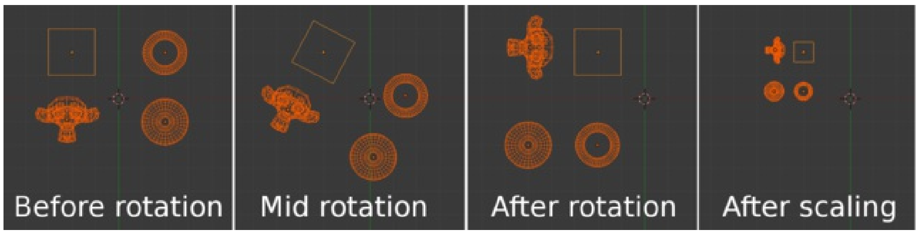
Read more about selecting different pivot points »



Display of active elements in Object mode is shown on the left of the image where the active element (cube) is a lighter orange. Active elements for vertices, edges and faces in Edit mode are displayed in white and are shown on the right.

In Object mode

When in Object mode, rotation and scaling happen around the active Object’s center. This is shown by the figure below where the active Object (the cube) remains in the same location (note its position relative to the 3D cursor) while the other Objects rotate and scale in relation to the active element.



Rotation and scaling with the cube as the active element.

In Edit mode

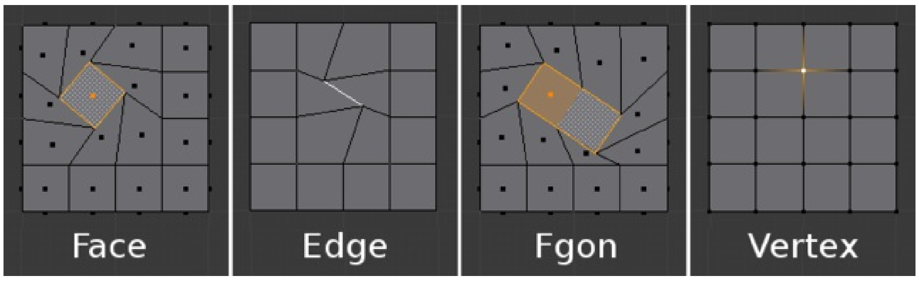
Using the active element as a pivot point in Edit mode may seem complex but all the possible transformations follow a few rules:

- The pivot point is always at the median of the active element(s).
- The transformations occur by transformation of the **vertices** of the selected element(s). If an un-selected element shares one or more vertices with a selected element, the un-selected element will undergo some degree of transformation as well.

Let’s examine the following examples: in either case we will see that these two discrepancies are applied.

Single selection

When one single element is selected it becomes automatically active. In the image below, you can see that when it is transformed its vertices move, with the consequence that any adjacent element which shares one or more vertices with the active element is also transformed.



Edit mode and only one element selected.

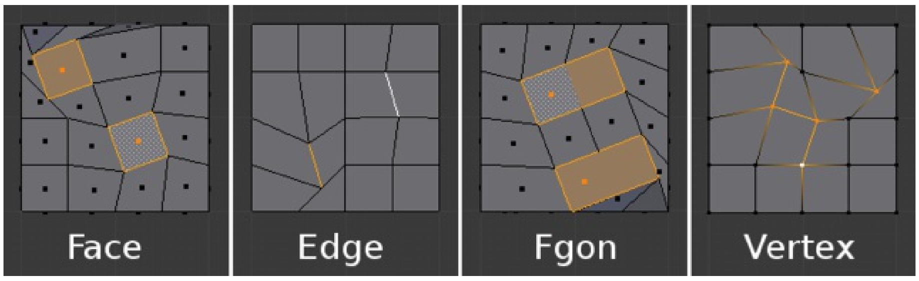
Let's review each case:

- *Faces* have their pivot point where their selection dot appears, which is where the median of their vertices is.

- *Edges* have their pivot point on their middle since this is always where the median of an edge is.
- *Fgons* behave the same as faces.
- A single *Vertex* has no dimensions at all so it can’t show any transformation (except translation, which is not affected by the pivot point).

Multiple selection

When multiple elements are selected they all transform. The pivot points stay in the same place as what we’ve seen above, with only one exception for Fgons. In the image below, the selected elements have been rotated.



Edit mode and multiple selections.

- For *Faces* the transformation occurs around the selection dot of the active face.
- *Edges* also keep the same behavior with their pivot point at their median.
- *Fgons* behave exactly like faces.
- There is a case for *Vertices* this time: the active Vertex is where the pivot point resides. All other vertices are transformed relative to it.

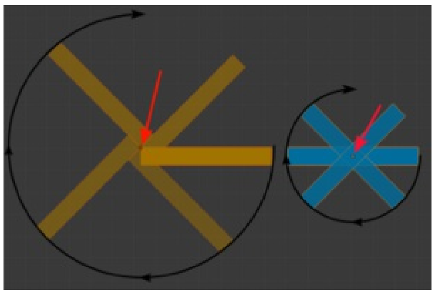
Individual Origins as Pivot

Mode: Object mode and Edit mode

Hotkey: Ctrl.

Menu: Select from the following icon in the 3D window header

In Object mode


Rotation around individual origins.

The Origin of an Object is shown in the 3D view by a small orange circle. This is highlighted in the image to the right by the red arrow. The origin tells Blender *where that Object is in 3D space*. What you see in the 3D view (vertices, edges etc) is what makes up the Object.

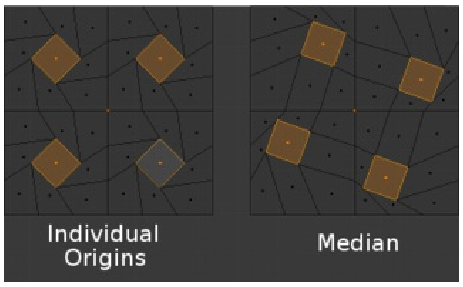
While the Origin is equivalent to the center of the *Object*, it does not have to be located in the center of the *Mesh*. This means that an Object can have its center located on one end of the mesh or even completely outside the mesh. For example, the orange rectangle in the image has its Origin located on the far left of the mesh.

Now let’s examine *Rotation around the individual origins*.

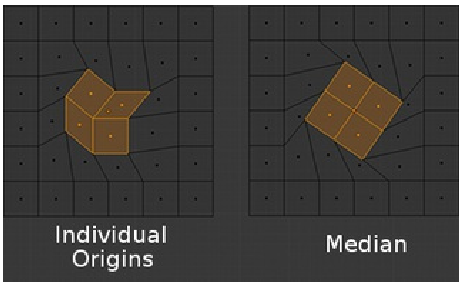
- The blue rectangle has its Origin located in the center of the mesh, while the orange rectangle has its Origin located on the left hand side.
- When the Pivot Point is set to Individual Origins, the center of each Object (indicated by the red arrow) remains in place while the Object rotates around it in the path shown by the black arrow.

In Edit mode

In Edit mode, setting the Pivot Point to Individual Origins produces different results when the selection mode is set to Vertex, Edge or Face. For example, Vertex mode produces results similar to setting the pivot point to median and Edge mode often produces distorted results. Using Individual Origins in Face mode produces the most predictable results.

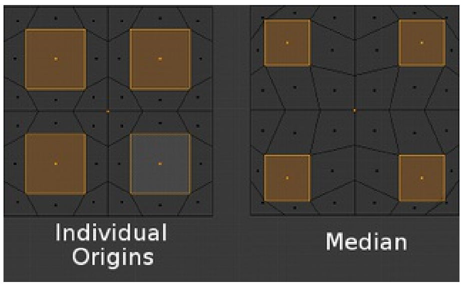


Rotation of individual faces with the pivot point indicated by the image text.

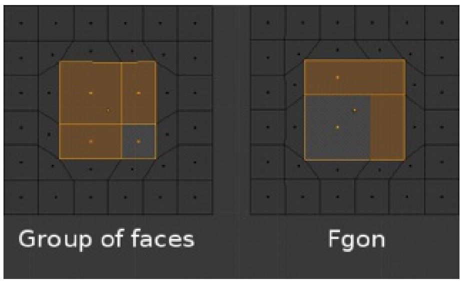


Rotation of grouped faces with the pivot point indicated by the image text.

As can be seen in the images above, faces that touch each other will deform when rotated when the pivot point is set to Individual Origins. Faces that do not touch will rotate around their Individual Origins (their center).

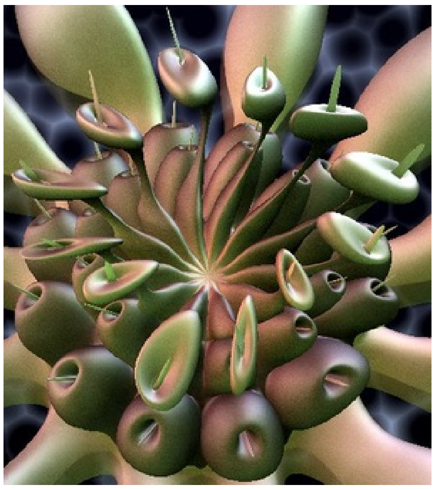


Scaling with *non-touching* faces.



Scaling with *touching* faces.

Groups of faces and Fgons can be scaled without their outside perimeter being deformed. However, the individual faces inside will not be scaled uniformly.


Modeling with faces and individual origins as the pivot point.

Once you are aware of its limitations and pitfalls, this tool can save a lot of time and lead to unique shapes. This “anemone” was modeled from a 12 sided cylinder in about 10 minutes by repeatedly using this workflow: extrusions of individual faces, scaling with *median as a pivot point*, and scaling and rotations of those faces with *Individual Origins as pivot points*.

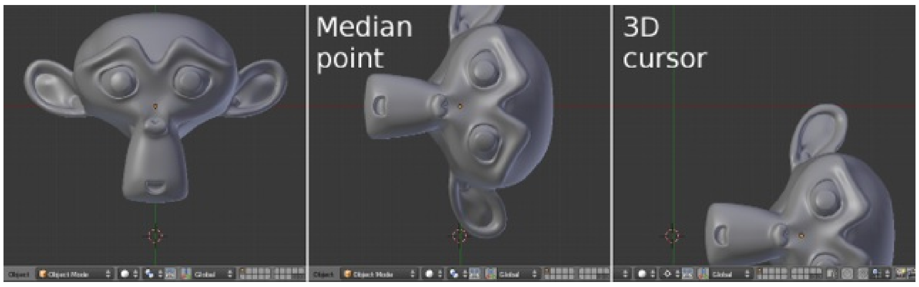
3D Cursor as Pivot

Mode: Object mode and Edit mode

Hotkey: .

The 3D cursor is the most intuitive of the pivot points. With the 3D cursor selected as the active pivot point (from either the Window Header or via the . hotkey), simply position the 3D cursor and then do the required transformation. All rotation and scaling transformations will now be done relative to the location of the 3D cursor. The image below shows the difference when rotating an Object from its starting position (first panel) 90 degrees around the median point (second panel) and 90 degrees around the 3D cursor (third panel).

Read more about selecting different pivot points »

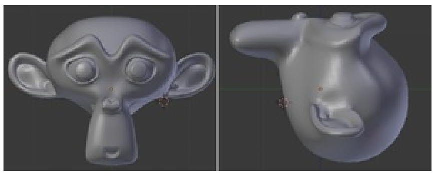


Rotation around the 3D cursor compared to the median point.

Positioning the 3D cursor

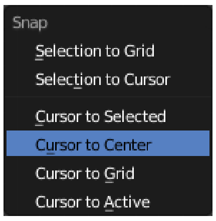
There are a few methods to position the 3D cursor.

Direct placement with the mouse


Positioning the 3D cursor with two orthogonal views.

- Using LMB in the 3D area will place the 3D cursor directly under your mouse pointer. For accuracy you should use two perpendicular orthogonal 3D views, i.e. any combination of top (7 NumPad), front (1 NumPad) and side (3 NumPad). That way you can control the positioning along two axes in one view and determine depth in the second view.

Using the Snap menu



The Snap menu (⇧ ShiftS or Object/Mesh » Snap) will allow you to snap the cursor in the following ways:

- Cursor to Selected: snaps the cursor to the currently selected vertex, edge or face. In Object mode this option will snap the cursor to the center of the currently selected Object.
- Cursor to Center: snaps the cursor to the origin point of the grid (location 0,0).
- Cursor to Grid: snaps the cursor to the nearest **visible** part of the grid.
- Cursor to Active: snaps the cursor to the *active* (last selected) object, edge, face or vertex.

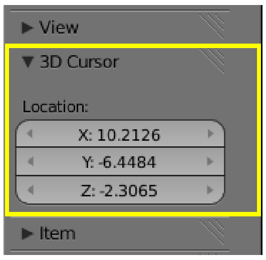
The Cursor to Selected option is also affected by the number of elements in the selection and the current pivot point. For example, with several elements selected and the Bounding Box Center pivot point active, the Cursor to Selected option will snap the 3D cursor to the:

- **Center of the bounding box** surrounding the objects' centers in Object mode or the **center of the bounding box** surrounding the selected vertices when in Edit mode.

When the Median Point pivot point is selected, Cursor to Selected will snap the 3D cursor to:

- The median of the object centers in Object mode and the median of the selected vertices in Edit mode.

Numeric input



The 3D Cursor panel of the Properties shelf.

The 3D cursor can also be positioned by entering Numeric location values into the 3D cursor panel of the Properties shelf (N).

Median Point as Pivot

Mode: Object mode and Edit mode

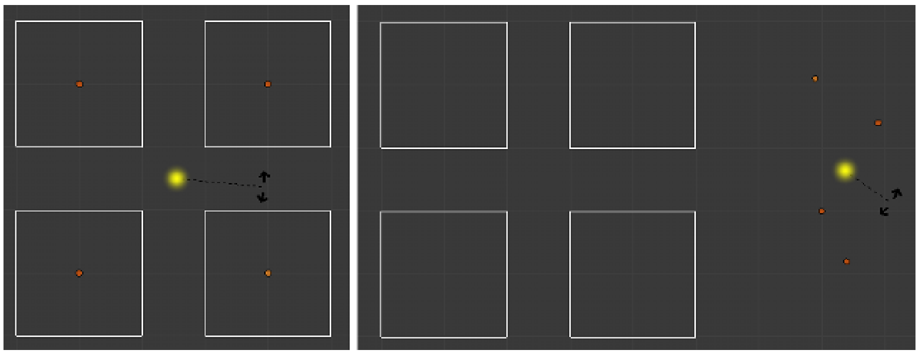
Hotkey: Ctrl,

Menu: Select from the following icon in the 3D window header

The Median Point can be considered to be broadly similar to the concept of Center of Gravity (COG). If we assume that every element (Object, face, vertex etc) of the selection has the same mass, the median point would sit at the point of equilibrium for the selection (the COG).

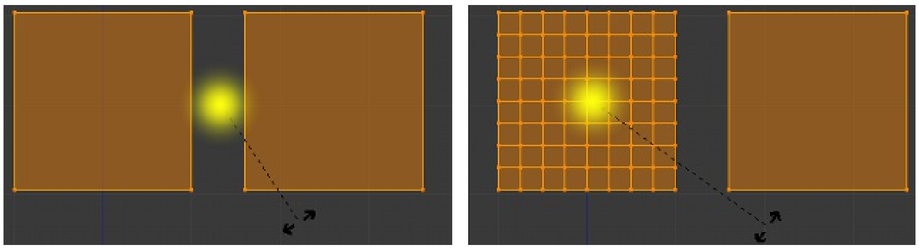
In Object Mode

In Object Mode, Blender only considers the Object centers when determining the median point. This can lead to some counterintuitive results. In the Object Mode median points image below, you can see that the median point is between the Object centers and can be nowhere near the Objects' mesh.


Median points in Object Mode. The Median point is indicated by the yellow dot.

In Edit Mode

In Edit Mode, the median point is determined via the part of the selection that has the most elements. For example, in the *Median points in Edit Mode* image, when there are two cubes with an equal number of vertices, the median point lies directly between the two cubes. However, if we subdivide one cube multiple times so that it has many more vertices, you can see that the median point has shifted to the region with the most vertices.


Median points in Edit Mode. The Median point is indicated by the yellow dot.

Bounding Box Center as Pivot

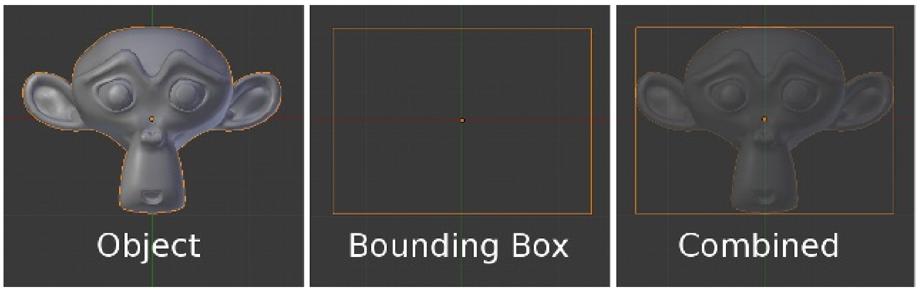
Mode: Object mode and Edit mode

Hotkey: ,

Menu: Select from the following icon in the 3D window header

The bounding box is a rectangular box that is wrapped as tightly as possible around the selection. It is oriented parallel to the world axes. In this mode the pivot point lies at the center of the bounding box. You can set the pivot point to bounding box with the , hotkey or via the menu in the Window Header. The image below shows how the Object's Bounding Box size is determined by the size of the Object.

Read more about selecting different pivot points »

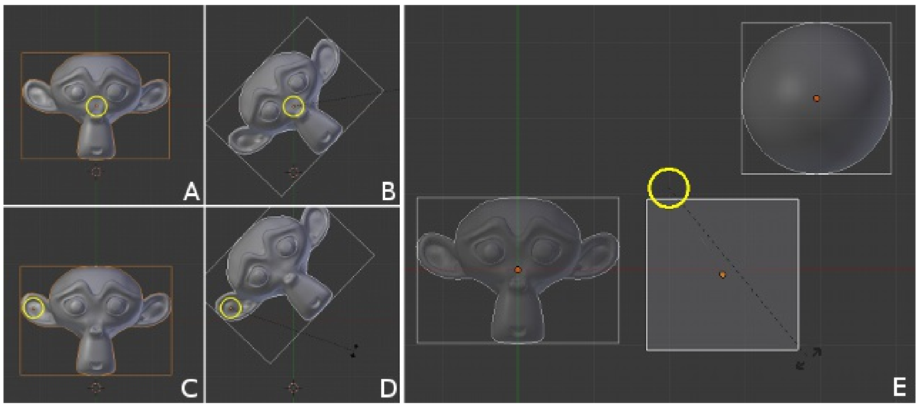


Relationship between an Object and its Bounding Box.

In Object mode

In Object mode, the bounding box is wrapped around the Object and transformation takes place relative to the location of the Object center (indicated by the yellow circle). The image below shows the results of using the Bounding Box as the pivot point in a number of situations.

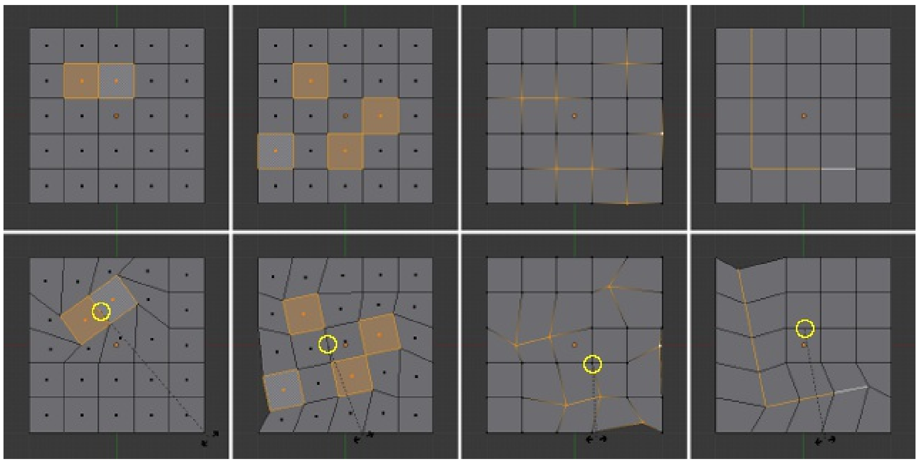
For example, images A (before rotation) and B show rotation when the Object center is in its default position, while images C (before rotation) and D shows the result when the Object center has been moved. Image E shows that when multiple Objects are selected, the pivot point is calculated based on the Bounding Box of all the selected Objects.



The grid of four images on the left (ABCD) shows the results of Object rotation when the pivot point is set to Bounding Box. The image to the right (E) shows the location of the Bounding Box pivot point when multiple Objects are selected. The pivot point is shown by a yellow circle.

In Edit mode

This time it is the ObData that is enclosed in the bounding box. The bounding box in Edit mode takes no account of the Object(s) centers, only the center of the selected vertices.


The effects of rotation in different mesh selection modes when the bounding box is set as the pivot point. The pivot point is shown by a yellow circle.

Snapping

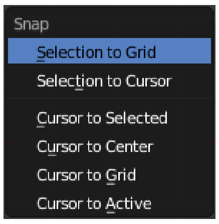
There are two types of snap operations that you can use in Blender. The first type snaps your selection or cursor to a given point while the second type is used during transformations (translate, rotate, scale) and snaps your selection to elements within the scene.

Snap

Mode: Object and Edit modes

Hotkey: ⇧ ShiftS

The Snap menu (also available from the 3D header in both Object and Edit mode (Object » Snap and Mesh » Snap). This menu provides a number of options to move the cursor or your selection to a defined point (the cursor, selection or the grid).



Snap menu

Selection to Grid
: Snaps the currently selected object(s) to the nearest grid point.

Selection to Cursor
: Snaps the currently selected object(s) to the cursor location.

Cursor to Selected
: Moves the cursor to the center of the selected object(s).

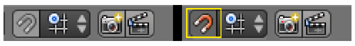
Cursor to Center
: Moves the cursor to the center of the grid.

Cursor to Grid
: Moves the cursor to the nearest grid point.

Cursor to Active
: Moves the cursor to the center of the active object.

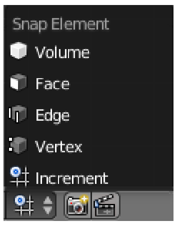
Transform Snapping

The ability to snap Objects and Mesh element to various types of scene elements during a transformation is available by toggling the magnet icon (which will turn red) in the 3D view's header buttons.


Magnet icon in the 3D view header (red when enabled).

Snapping Modes

Snap Element



Snap Element menu

Volume
: Snaps to regions within the volume of the first Object found below the mouse cursor. Unlike the other options, this one controls the depth (i.e. Z-coordinates in current view space) of the transformed element. By toggling the button that appears to the right of the snap target menu (see below), target objects will be considered as a whole when determining the volume center.

Face
: Snap to the surfaces of faces in mesh objects. Useful for retopologizing.

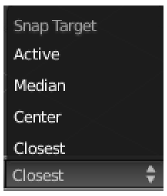
Edge
: Snap to edges of mesh objects.

Vertex
: Snap to vertices of mesh objects.

Increment
: Snap to grid points. When in Orthographic view, the snapping increment changes depending on zoom level. Please note: in this

context the grid does not mean the visual grid cue displayed. Snapping will use the resolution of the displayed grid, but all transformations are relative to the initial position (before the snap operation).

Snap Target



Snap Target menu.

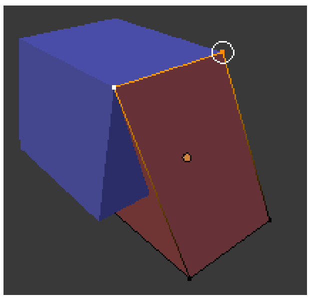
Snap target options become active when either Vertex, Edge, Face, or Volume is selected as the snap element. These determine what part of the selection snaps to the target objects.

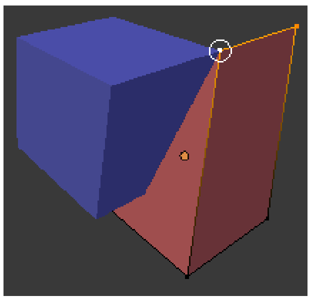
Active
: move the active element (vertex in Edit mode, object in Object mode) to the target.

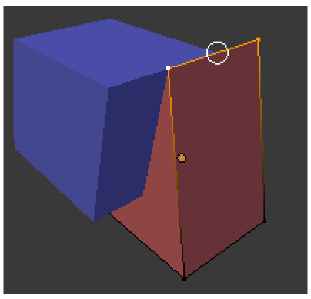
Median
: move the median of the selection to the target.

Center
: move the current transformation center to the target. Can be used with 3D cursor to snap with an offset.

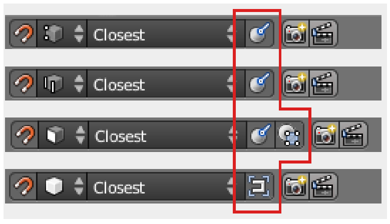
Closest
: move the closest point of the selection to the target.


Closest

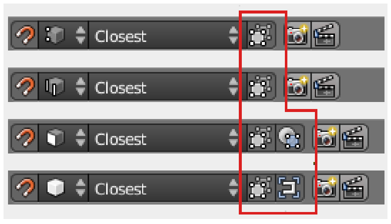

Active


Median

Additional snap options



Object mode



Edit mode

As seen by the red highlighted areas in the image above, additional controls are available to alter snap behaviour. These options vary between mode (Object and Edit) as well as Snap Element. The four options available are:

- Align rotation with the snapping target.
- Project individual elements on the surface of other objects.
- Snaps elements to its own mesh.
- Consider Objects as whole when finding volume center.

Snapping

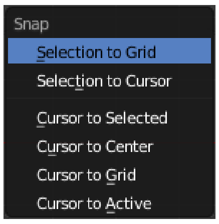
There are two types of snap operations that you can use in Blender. The first type snaps your selection or cursor to a given point while the second type is used during transformations (translate, rotate, scale) and snaps your selection to elements within the scene.

Snap

Mode: Object and Edit modes

Hotkey: ⇧ ShiftS

The Snap menu (also available from the 3D header in both Object and Edit mode (Object » Snap and Mesh » Snap). This menu provides a number of options to move the cursor or your selection to a defined point (the cursor, selection or the grid).



Snap menu

Selection to Grid
: Snaps the currently selected object(s) to the nearest grid point.

Selection to Cursor
: Snaps the currently selected object(s) to the cursor location.

Cursor to Selected
: Moves the cursor to the center of the selected object(s).

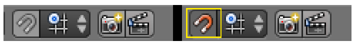
Cursor to Center
: Moves the cursor to the center of the grid.

Cursor to Grid
: Moves the cursor to the nearest grid point.

Cursor to Active
: Moves the cursor to the center of the active object.

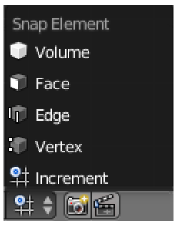
Transform Snapping

The ability to snap Objects and Mesh element to various types of scene elements during a transformation is available by toggling the magnet icon (which will turn red) in the 3D view's header buttons.


Magnet icon in the 3D view header (red when enabled).

Snapping Modes

Snap Element



Snap Element menu

Volume
: Snaps to regions within the volume of the first Object found below the mouse cursor. Unlike the other options, this one controls the depth (i.e. Z-coordinates in current view space) of the transformed element. By toggling the button that appears to the right of the snap target menu (see below), target objects will be considered as a whole when determining the volume center.

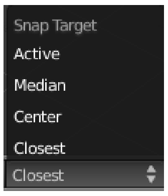
Face
: Snap to the surfaces of faces in mesh objects. Useful for retopologizing.

Edge
: Snap to edges of mesh objects.

Vertex
: Snap to vertices of mesh objects.

Increment
: Snap to grid points. When in Orthographic view, the snapping increment changes depending on zoom level.

Snap Target



Snap Target menu.

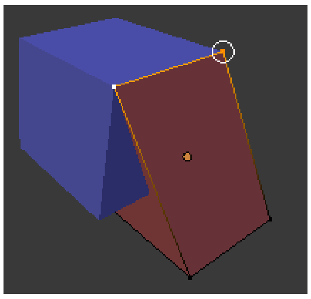
Snap target options become active when either Vertex, Edge, Face, or Volume is selected as the snap element. These determine what part of the selection snaps to the target objects.

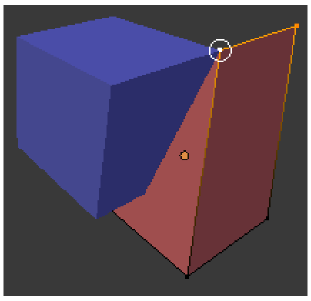
Active
: move the active element (vertex in Edit mode, object in Object mode) to the target.

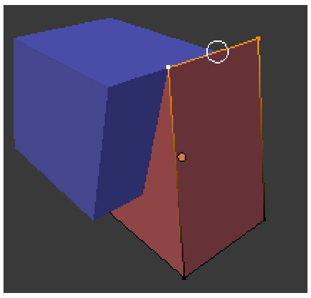
Median
: move the median of the selection to the target.

Center
: move the current transformation center to the target. Can be used with 3D cursor to snap with an offset.

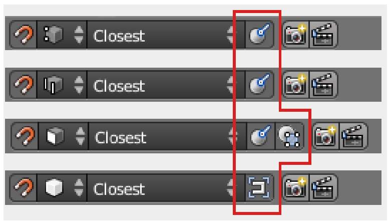
Closest
: move the closest point of the selection to the target.


Closest

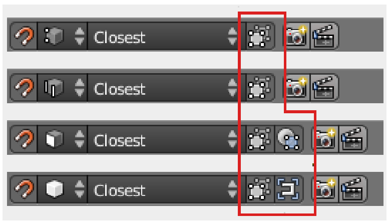

Active


Median

Additional snap options



Object mode

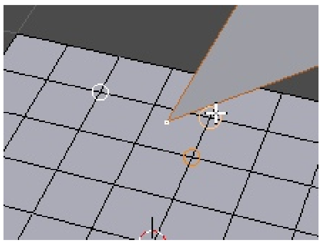


Edit mode

As seen by the red highlighted areas in the image above, additional controls are available to alter snap behaviour. These options vary between mode (Object and Edit) as well as Snap Element. The four options available are:

- Align rotation with the snapping target.
- Project individual elements on the surface of other objects.
- Snaps elements to its own mesh.
- Consider Objects as whole when finding volume center.

Multiple Snap Targets


Multiple snapping targets.

Once transforming a selection with Snapping on (not just with the Ctrl key held), you can press A to mark the current snapping point, then proceed to mark as many other snapping points as you wish and the selection will be snapped to the average location of all the marked points.

Marking a point more than once will give it more weight in the averaged location.

Proportional Edit

Proportional Edit is a way of transforming selected elements (such as vertices) while having that transformation affect other nearby elements. For example, having the movement of a single vertex cause the movement of unselected vertices within a given range. Unselected vertices that are closer to the selected vertex will move more than those farther from it (i.e. they will move proportionally relative to the location of the selected element). Since proportional editing affects the nearby geometry, it is very useful when you need to smoothly deform the surface of a dense mesh.

Sculpting
Blender also has Sculpt Mode that contains brushes and tools for proportionally editing a mesh without seeing the individual vertices.

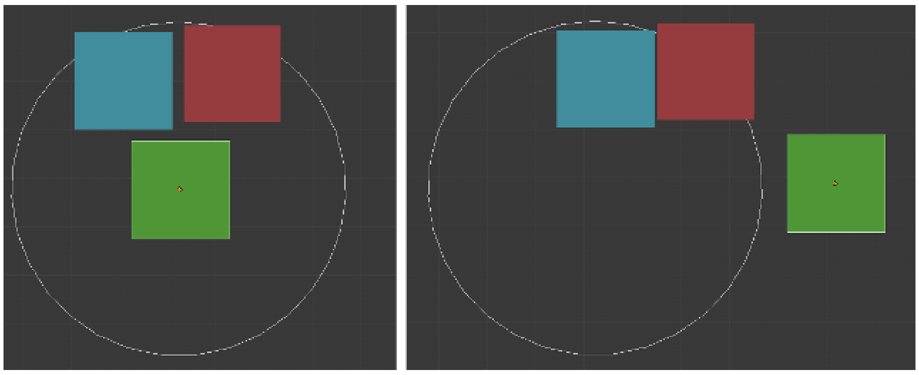
Object mode

Mode: Object mode

Hotkey: O

Menu: Via the icon in the header indicated by the yellow square in the below image.

Proportional editing is typically used in Edit mode, however, it can also be used in Object mode. In Object mode the tool works on entire objects rather than individual mesh components. In the image below, the green cube is the active Object, while the red and blue cubes are located within the proportional edit tool's radius of influence. When the green cube is moved to the right, the other two cubes follow the movement.


Proportional editing in Object mode.

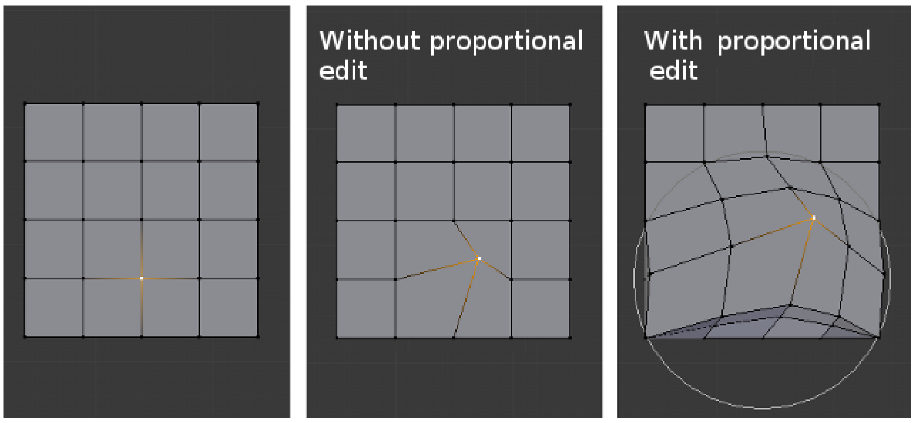
Edit mode

Mode: Edit mode

Hotkey: O / AltO / ⇧ ShiftO

Menu: Mesh » Proportional Editing and via the highlighted icon in the below image

When working with dense geometry, it can become difficult to make subtle adjustments to the vertices without causing visible lumps and creases in the model’s surface. When you face situations like this the proportional editing tool can be used to smoothly deform the surface of the model. This is done by the tool's automatic modification of unselected vertices within a given range.

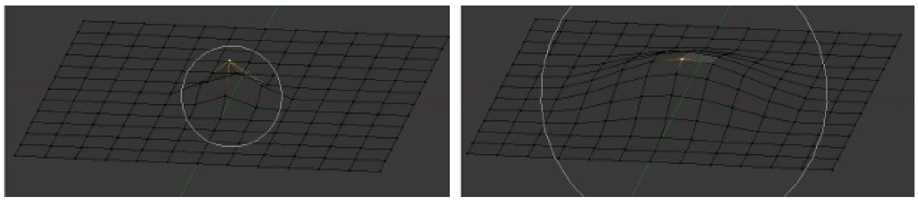


Proportional editing in Edit mode.

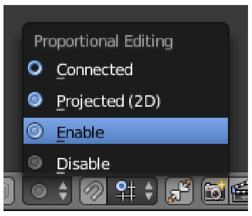
Influence

You can increase or decrease the radius of the proportional editing influence with the mouse wheel WheelUp / WheelDown or

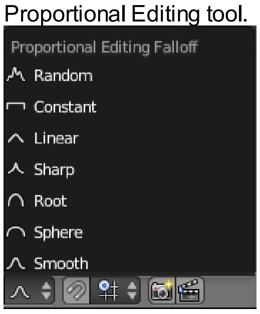
PageUp/PageDown respectively. As you change the radius, the points surrounding your selection will adjust their positions accordingly.


Influence circle.

Options



Proportional Editing tool.



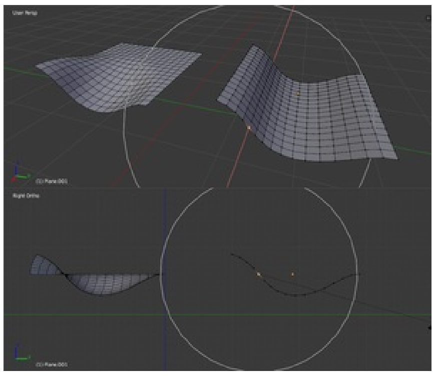
Falloff menu.

The Proportional Editing mode menu is on the 3D View header.

Disable (O or AltO)
: Proportional Editing is Off, only selected vertices will be affected.

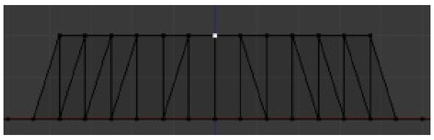
Enable (O or AltO)
: Vertices other than the selected vertex are affected, within a defined radius.

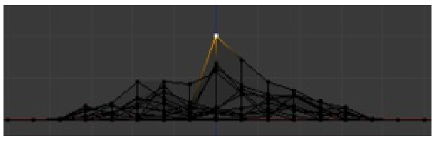
Projected (2D)
: Depth along the view is ignored when applying the radius.

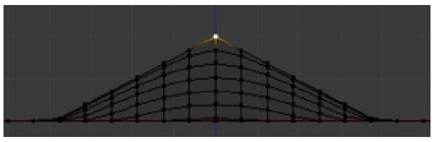

The difference between regular and Projected (2D) proportional option (right).

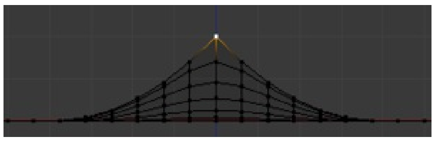
Connected (AltO)
: Rather than using a radius only, the proportional falloff spreads via connected geometry. This means that you can proportionally edit the vertices in a finger of a hand without affecting the other fingers. While the other vertices are physically close (in 3D space), they are far away following the topological edge connections of the mesh. The icon will have a grey center when Connected is active. This mode is only available in Edit mode.

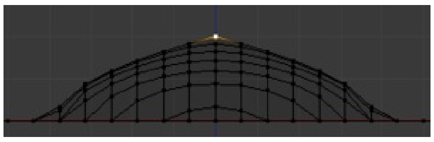
Falloff
: While you are editing, you can change the curve profile used by either using the Mesh » Proportional Falloff submenu, using the toolbar icon (*Falloff menu*), or by pressing ⇧ ShiftO to toggle between the various options.

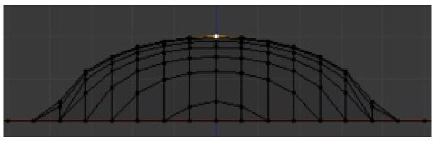

Constant, No Falloff.

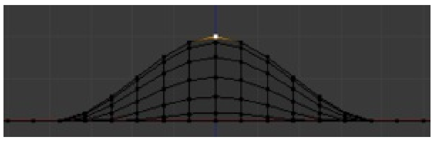

Random Falloff.


Linear Falloff.


Sharp Falloff.


Root Falloff.

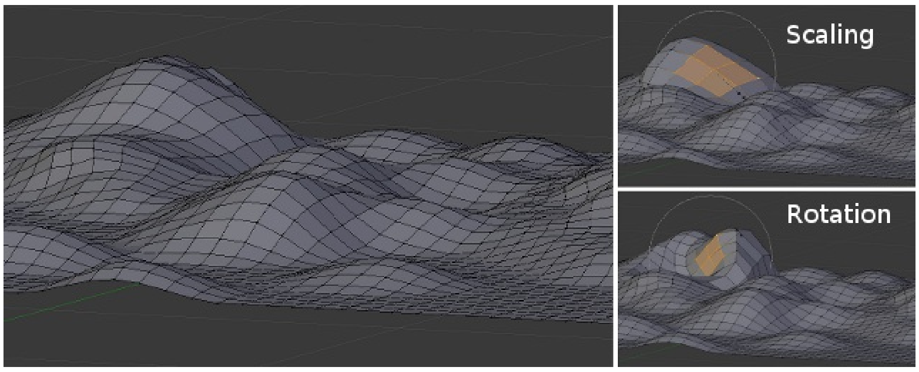

Sphere Falloff.


Smooth Falloff.

Examples

Switch to a front view (1 NumPad) and activate the grab tool with G. As you drag the point upwards, notice how nearby vertices are dragged along with it. When you are satisfied with the placement, click LMB to fix the position. If you are not satisfied, cancel the operation and revert your mesh to the way it looked before with RMB (or Esc).

You can use the proportional editing tool to produce great effects with the scaling (S) and rotation (R) tools, as *A landscape obtained via proportional editing* shows.



A landscape obtained via proportional editing.

Combine these techniques with vertex painting to create fantastic landscapes. The *final rendered landscape* image below shows the results of proportional editing after the application of textures and lighting.

Final rendered landscape.

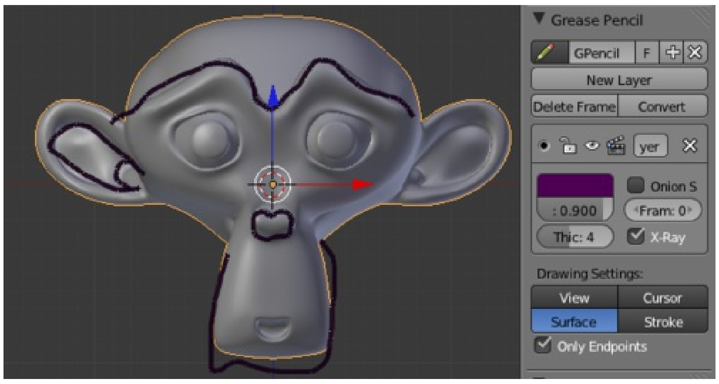
Viewport Sketching

The Grease Pencil is a tool that allows you to draw sketches and annotations in the 3D view and UV/Image editor windows using freehand strokes. It is a tool that can be linked back to traditional 2D paper and pencil workflows where rough “guideline” sketches were often used for planning and quickly communicating ideas. In digital fields, a similar workflow was used by drawing on monitors with wax crayons and pencils. It is often useful to be able to directly scribble on to a work in progress instead of having to do so in a separate place (i.e. another part of the window, or even in a different application altogether).

The name “Grease Pencil” is derived in homage to the wax crayons/pencils that early CG Animators used to draw arcs and other planning notes on their CRT’s with.

In addition to uses for animators in planning their poses and motion curves, Grease Pencil can also be useful in a number of scenarios, including but not limited to:

- Planning topology and/or layout of models.
- Director’s shot review tool.
- “Whiteboard” and assignment review tool for educators.



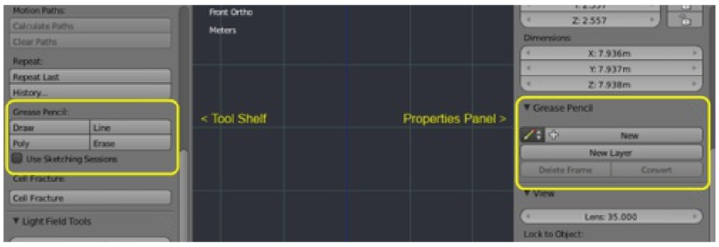
The Grease Pencil in action.

The next few pages explain how to use this tool:

- Drawing sketches.
- Layers and Animation.
- Converting sketches to geometry.

Drawing With Grease Pencil

1. Enable the Grease Pencil by clicking Draw, Line, Poly or Erase from the Toolshelf (T). A new layer will be automatically added for you to draw on.
2. A new layer can be added from the Grease Pencil Properties panel. This panel can also be used to customise the color, opacity and thickness of the pencil lines. Changes to these settings will affect all strokes on the current layer.



Grease Pencil Tool Shelf and Properties Panel.

Grease Pencil sketches can be converted to editable geometry and used to aid the animation process.

- Read more about Layers and Animation »
- Read more about Converting sketches to geometry »

Drawing

The Toolshelf provides a number of options for drawing with the Grease Pencil which are detailed below. The Toolshelf can be seen in the screenshot Grease Pencil Tool Shelf and Properties Panel above.

Grease Pencil Mode and Shortcut Summary

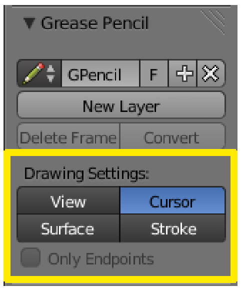
Type	Shortcut	Usage
Draw	D LMB	Draw a new stroke (multiple short, connected lines). The stroke will finish when you release the mouse button.
Line	CtrlD LMB	Draw a new line in rubber band mode. The line will finish when you release the mouse button.
Poly	CtrlD RMB	Draw connected lines by clicking at various points. Lines will be automatically added to connect the two points.
Erase	D RMB	Erases segments of strokes that fall within the radius of the eraser “brush”. The erasing will continue until the mouse button is released. If begun with Erase, either RMB or LMB will erase strokes. The size of the eraser “brush” can be controlled with Wheel or + NumPad and - NumPad keys (while still holding RMB).

Sketching Sessions

A Sketching Session allows for rapid sketching with the Grease Pencil when multiple strokes are desired. With this option set, a sketching session starts when a Grease Pencil stroke is made. The type of session (Draw, Line, Poly, Erase) is determined by the first stroke made which can be done via hotkeys or the Toolshelf. Use Esc or ↵ Enter to exit the sketching session. Note that in a Erase Sketching Session both LMB or RMB can be used once the session has started.

Shared Grease Pencil Settings

Drawing Settings



Grease Pencil Drawing Settings.

In the Grease Pencil Panel of the Properties shelf (N) there are several choices for Drawing Settings.

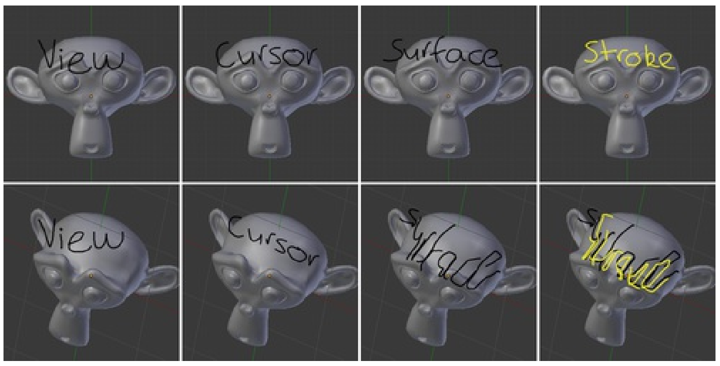
View
: New strokes are locked to the view.

Cursor (3D view only)
: New strokes are drawn in 3D-space, with position determined by the 3D cursor and the view rotation at the time of drawing. Cursor is available as an option in the UV/Image Editor but it functions identically to the View option.

Surface (3D view only)
: New strokes are drawn in 3D-space, with their position projected onto the first visible surface.

Stroke (3D view only)
: New strokes are drawn in 3D-space, with their position projected onto existing visible strokes. Note that strokes created with View are not in 3D-space and are not considered for this projection.

Enabling the Only Endpoints setting applies the drawing setting only to the endpoints of the stroke. The part of the stroke between the endpoints is adjusted to lie on a plane passing through the endpoints.



The effect of different Drawing Settings on Grease Pencil strokes.

Sensitivity When Drawing

The default settings for the sensitivity of mouse/stylus movement when drawing have been set to reduce jitter while still allowing fine movement. However, if these are not appropriate they can be altered in User Preferences window » Editing » Grease Pencil.

Manhattan Distance
: The minimum number of pixels the mouse should have moved either horizontally or vertically before the movement is recorded. Decreasing this should work better for curvy lines.

Euclidean Distance
: The minimum distance that the mouse should have traveled before movement is recorded.

Eraser Radius
: The size of the eraser “brush”.

Smooth Stroke
: This turns on the post-processing step of smoothing the stroke to remove jitter. It is only relevant when not drawing straight lines. By default this is enabled. It should be noted that it can often cause “shrinking” of drawings, and may be turned off if the results are not desirable.

Simplify Stroke
: This turns on the post-processing step of simplifying the stroke to remove about half of current points in it. It is only relevant when not drawing straight lines. By default this is disabled. As with Smooth Stroke, it can often cause “shrinking” of drawings, and loss of precision, accuracy and smoothness.

Additional Notes For Tablet Users

- The thickness of a stroke at a particular point is affected by the pressure used when drawing that part of the stroke.
- The “eraser” end of the stylus can be used to erase strokes.

Layers

Grease Pencil sketches are organized in layers, much like those you could find in the GIMP or Photoshop. These layers are not related to any of the other layer systems in Blender, and also do not have an upper limit on the maximum number of layers that can be used. Like the layers in the aforementioned applications, these layers can also be renamed, locked, hidden, and deleted.

Their main purpose is to collect together a bunch of sketches that belong together in some meaningful way (i.e. “blocking notes”, “director’s comments on blocking”, or “guidelines”). For this reason, all the strokes on a layer (not just those made after a particular change) are affected by that layer’s color, opacity, and stroke thickness settings.

By default, most operations occur only on the *active* layer. The active layer can be identified as the one with the different panel color (in the default set, a light orangy-brown color). Clicking on a layer, or changing any of its settings will make it the new active layer.

The active layer can also be identified by looking at the status indicator (in the top right-hand corner of every view with Grease Pencil data being shown).

Animation of the Sketches

Grease Pencil can be used to do basic pencil tests (i.e. 2D animation in flipbook style). Sketches are stored on the frame that they were drawn on, as a separate drawing (only on the layer that they exist on). Each drawing is visible until the next drawing for that layer is encountered. The only exception to this is the first drawing for a layer, which will also be visible before the frame it was drawn on.

Therefore, it is simple to make a pencil-test/series of animated sketches:

1. Go to first relevant frame. Draw.
2. Jump to next relevant frame. Draw some more.
3. Keep repeating process, and drawing until satisfied. Voila! Animated sketches.

Onion Skinning

Onion-skinning (also known as ghosting), is a useful tool for animators, as neighboring frame(s) are lightly drawn by Blender. It allows animators to make judgments about movements, by comparing movement from different frames.

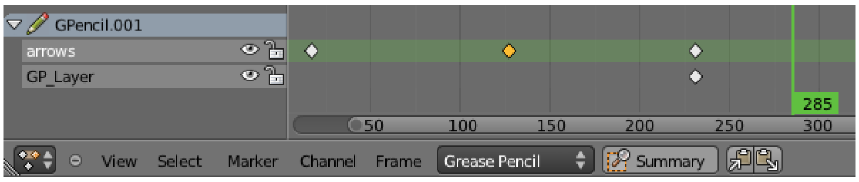
Usage Notes:

- Onion-skinning is enabled per layer by clicking on the Onion Skin button in the grease pencil properties panel.
- The Frames field, directly under the Onion Skin button, controls how many frames will be drawn. When Frames is **0**, only the drawing on either side of the current frame will be visible. Otherwise, this field specifies the maximum number of frames on either side of the current frame that will result in a neighboring drawing.

Adjusting Timing of Sketches

It is possible to set a Grease-Pencil block to be loaded up in the DopeSheet for editing of the timings of the drawings. This is especially useful for animators blocking out shots, where the ability to re-time blocking is one of the main purposes of the whole exercise.

1. In an Dope Sheet window, change the mode selector (found beside the menus) to Grease Pencil (by default, it should be set to DopeSheet).
2. At this point, the DopeSheet should now display a few “channels” with some “keyframes” on them. These “channels” are the layers, and the “keyframes” are the frames at which the layer has a sketch defined. They can be manipulated like any other data in the DopeSheet can be.



All the available Grease-Pencil blocks for the current screen layout will be shown. The Area/Grease-Pencil datablocks are drawn as green channels, and are named with relevant info from the views. They are also labeled with the area (i.e. window) index (which is currently not shown anywhere else though).

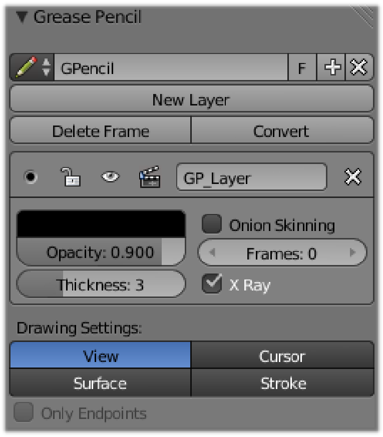
Copying Sketches

It is possible to copy sketches from a layer/layers to other layers in the Action Editor, using the “Copy”/“Paste” buttons in the header. This works in a similar way as the copy/paste tools for keyframes in the Action Editor.

Sketches can also be copied from one screen (or view) to another using these tools. It is important to keep in mind that keyframes will only be pasted into selected layers, so layers will need to be created for the destination areas too.

Converting Sketches to Objects

In the 3D view, sketches on the active layer can be converted to geometry, based on the current view settings, by transforming the points recorded when drawing (which make up the strokes) into 3D-space. Currently, all points will be used, so it may be necessary to simplify or subdivide parts of the created geometry for standard use.



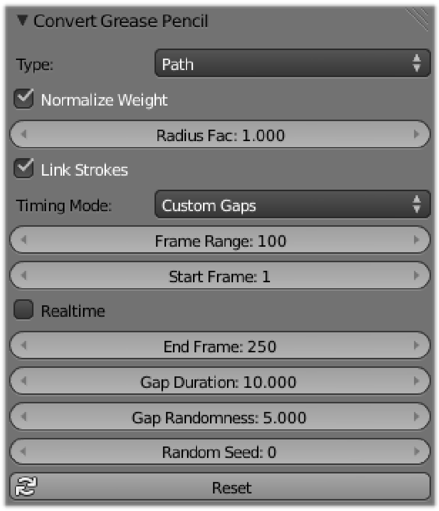
Sketches can currently be converted into curves, as proposed by the Convert Grease Pencil menu popped-up by the Convert button in the grease pencil properties

- Path will create NURBS 3D curves of order 2 (i.e. behaving like polylines).
- Bezier Curve will create Bezier curves, with free “aligned” handles (i.e. also behaving like polylines).

Why “polyline-like” curves?
To get by default curves following exactly the grease pencil strokes. If you want a smoothed curve, just edit it to get auto handles (for Bezier), or raise its order (for NURBS).

Converting to Mesh
If you want to convert your sketch to a mesh, simply choose first NURBS, and then convert the created curve to a mesh…

General Options



Stroke’s width will be used to set the curve’s control points’ radii and weights (**not** NURBS weights, but those used e.g. as goal by the softbody simulation…). The default behavior is to get strokes’ width (as defined in its settings – and which might have been modulated by the pen pressure), to multiply it by a given constant (0.1), and to assign it directly to weights. Radii get the same value scaled by the Radius Fac factor (e.g. with a **10.0** factor, a stroke width of **3** will give radii of **3.0**…).

Normalize Weight (enabled by default) will scale weights value so that they tightly fit into the `[0.0, 1.0]` range.

All this means that with a pressure tablet, you can directly control the radius and weight of the created curve, which can affect e.g. the width of an extrusion, or the size of an object through a Follow Path constraint or Curve modifier!

Link Strokes (enabled by default) will create a single spline (i.e. curve element) from all strokes in active grease pencil layer. This especially useful if you want to use the curve as a path. All the strokes are linked in the curve by “zero weights/radii” sections.

Timing

Grease pencil now stores “dynamic” data, i.e. how fast they were drawn. When converting to curve, those data can be used to create an Evaluate Time F-Curve (in other words, a path animation), that can be used e.g. to control another object’s position along that curve (Follow Path constraint, or, trough a driver, Curve modifier). So this allows you to reproduce your drawing movements.

All those “timing” options need Link Stroke to be enabled – else they would not make much sense!

Please note that if you use this tool with older grease pencil's strokes (i.e. some without any timing data), you will only have a subset of those options available (namely, only linear progression along the curve over a specified range of frames…).

Timing Mode
: This control let you choose how timing data are used.

  No Timing
  : Just create the curve, without any animation data (hence all following options will be hidden)…

  Linear
  : The path animation will be a linear one.

  Original
  : The path animation will reflect to original timing, including for the “gaps” (i.e. time between strokes drawing).

  Custom Gaps
  : The path animation will reflect to original timing, but the “gaps” will get custom values. This is especially useful if you have very large pauses between some of your strokes, and would rather like to have “reasonable” ones!

Frame Range
: The “length” of the created path animation, in frames. In other words, the highest value of Evaluation Time.

Start Frame
: The starting frame of the path animation.

Realtime
: When enabled, the path animation will last exactly the same duration it took you do draw the strokes.

End Frame
: When Realtime is disabled, this defines the end frame of the path animation. This means that the drawing timing will be scaled up or down to fit into the specified range.

Gap Duration
: Custom Gaps only. The average duration (in frames) of each gap between actual strokes. Please note that the value entered here will only be exact if Realtime is enabled, else it will be scaled, exactly as the actual strokes’ timing is!

Gap Randomness
: Only when Gap Duration is non-null. The number of frames actual gap duration can vary of. This allows the creation of gaps having an average well defined duration, yet keeping some random variations to avoid an “always the same” effect.

Random Seed
: The seed fed to the random generator managing gaps duration variations. Change it to get another set of gaps duration in the path animation.

Example

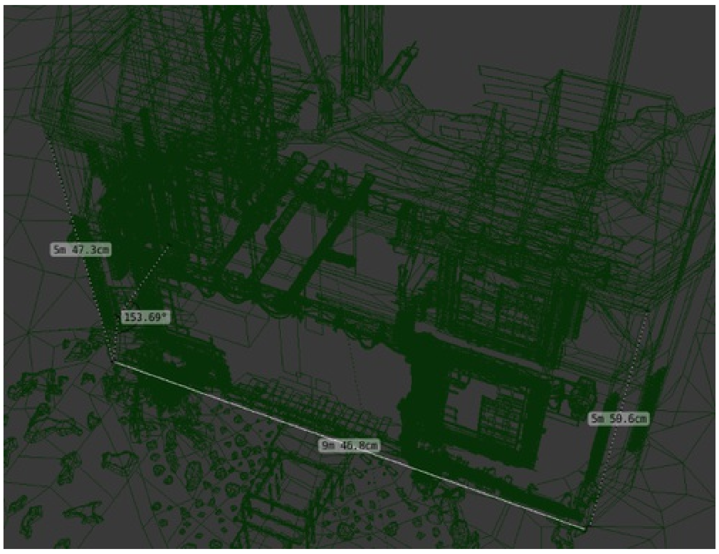
Here is a simple “hand writing” video created with curves converted from sketch data:

[video link]

And the blend file : File:ManGreasePencilConvertToCurveDynamicExample.blend

Ruler and Protractor

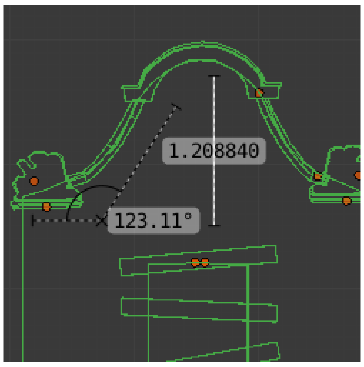
The ruler can be accessed from the toolshelf, once activated you can use the ruler to measure lengths and angles in the scene.



Example of the ruler and protractor.

6mm 821µm
3mm 435.5µm
6mm 44.1µm
9mm 701.8µm
Example using the ruler to measure thickness.

Usage



Here are common steps for using the ruler.

- Activate the Ruler from the toolshelf.
- Click and drag in the viewport to define the initial start/end point for the ruler.
- Orbit the view and click on either end of the ruler to re-position it. Holding Ctrl enables snap to elements.
- Click on the middle to measure angles.
- Press Enter to store the ruler for later use or Esc to cancel.

Note:

- Editing operations can be used while the ruler is running, however tools like the knife can't be used at the same time.
- Unit settings and scale from the scene are used for displaying dimensions.

Shortcuts

- Ctrl LMB Adds new ruler.
- LMB Drag end-points to place them, Hold Ctrl to snap, Hold Shift to measure thickness.
- LMB Drag center-point to measure angles, drag out of the view to convert back to a ruler.
- Delete Deletes the ruler.
- CtrlC Copies the rulers value to the clipboard.
- Esc Exits
- Return Saves the rulers for the next time the tool is activated.

Modeling in Blender

As you have seen in the Quick Start chapter, the creation of a 3D scene needs at least three key things: Models, Materials and Lights. In this Part we will delve deeper into the first of these issues: modeling. Modeling is the art and science of creating a surface that mimics the shape of a real-world object or fits your imagination of abstract objects.

Objects come in many forms, shapes and sizes, so Blender has many different tools available to help you make your model quickly and efficiently:

Objects
: Working with objects as a whole

Meshes
: Working with the mesh that defines the shape of an object

Curves
: Using Curves to model and control objects

Surfaces
: Modeling a NURBS surface

Text
: Textual tools for putting words in 3D space

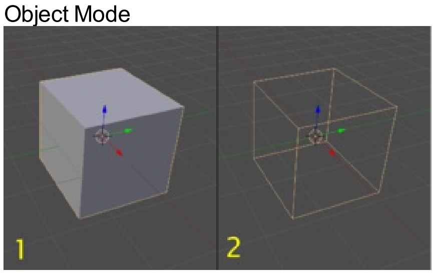
Meta Objects
: Globs and Globules

Duplications
: Duplicating Objects

Modeling Scripts
: Since Blender functionality is extensible via Python, there are a number of very useful scripts that assist you in modeling.

Many people use "box modeling" which starts with a basic cube, and proceeds with extruding faces and moving vertices to create a larger, more complicated mesh. For flat objects, like walls and table tops, you can use "curve modeling" which defines the outline using bezier or Nurbs curves, and then extrudes it to the desired thickness. Either method is fully supported in Blender using its modeling tools.

Object Mode



Selected object in 3D View: 1 - Solid shading, 2 - Wireframe shading.

The geometry of a scene is constructed from one or more Objects: For example Lamps, Curves, Surfaces, Cameras, Meshes, and the basic objects ("primitives") described in "Mesh Primitives".

Types of Objects

Meshes	Meshes are objects composed of Polygonal Faces, Edges and/or Vertices, and can be edited extensively with Blender's mesh editing tools.
Curves	Curves are mathematically defined objects, which can be manipulated with control handles or control points, rather than vertices.
Surfaces	Surfaces are four-sided patches that are also manipulated with control points. These are useful for very organic, and rounded, but simple forms.
Meta Objects	Meta Objects (or Metaballs) are objects formed by a function defining the 3d volume in which the object exists. Metaballs can create "Blobby" forms that have a liquid-like quality, when two or more Metaballs are used.
Text	Text objects create a 2d representation of a string of characters.
Armatures	Armatures are used for rigging 3d models in order to make them poseable and animateable.
Lattice	Nonrenderable wireframe. Commonly used for taking additional control over other objects with help of modifiers.
Empty	Empties are null objects that are simple visual transform nodes that do not render. They are useful for controlling the position or movement of other objects.
Speaker	Brings to scene source of sound.
Cameras	This is the virtual camera that is used to determine what appears in the render.
Lamps	These are used to place light sources in the scene.
Force Fields	Force fields are used in physical simulations. They give simulations external forces, creating movement, and are represented in 3d editor by small control objects.

Object Mode button.

Each object can be moved, rotated and scaled in Object Mode (see picture). However, not all of these transformations have an effect on all objects. For example, scaling a force field will not increase its effect.

Edit Mode button.

For making other changes to the geometry of editable objects, you should use Edit mode (see picture).

Once you've added a basic object, you remain in Object Mode. In earlier versions of Blender, you were automatically switched into Edit mode if the Object was a Mesh, a Curve or a Surface.

You can switch between Object Mode and Edit Mode by pressing ⇆ Tab.

The object's wireframe should now appear orange. This means that the object is now selected and active (see picture *Selected object*).

The (*Selected object*) image shows both the solid view and wireframe view of the default cube. To switch between wireframe and solid view, press Z.

Object Centers

Each object has an origin point. The location of this point determines where the object is located in 3D space. When an object is selected, a small circle appears, denoting the origin point. The location of the origin point is important when translating, rotating or scaling an object. See Pivot Points for more.

Moving Object Centers

Object Centers can be moved to different positions through 3D View window → Transform → Origin (press T to open panel):

- Geometry to Origin

  Move model to origin and this way origin of the object will also be at the center of the object.

- Origin to Geometry

  Move origin to the center of the object and this way origin of the object will also be at the center of the object.

- Origin to 3D Cursor

  Move origin of the model to the place of the 3D cursor.

- Origin to Center of Mass

  Move origin to calculated center of mass of model.

Erase Objects

Mode: Edit or Object mode

Hotkey: X or Del

Menu: Object → Delete

Erases or deletes selected objects.

Join Objects

Mode: Object mode

Hotkey: CtrlJ

Menu: Object → Join Objects

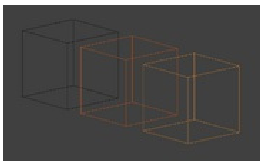
Joins all selected objects to one single object. Must be of the same type. Origin point is obtained from the previously *active* object. Performing a join is equivalent to adding new objects while in Edit mode. The non-active objects are deleted. Only the active object remains. This only works with editable objects, like meshes and curves.

Introduction

Selection determines which elements will be the target of our actions. Blender has advanced selection methods. Both in Object mode and in Edit mode.

Selections and the Active Object

Blender distinguishes between two different states of selection:


Unselected object in black, selected object in orange, and active object in yellow

- In Object mode the last (de)selected item is called the “Active Object” and is outlined in yellow (the others are orange). There is exactly one active object at any time (even when nothing is selected).

  Many actions in Blender use the active object as a reference (for example linking operations). If you already have a selection and need to make a different object the active one, simply re-select it with ⇧ Shift RMB.
- All other selected objects are just selected. You can select any number of objects.

Point Selection

The simplest form of object *selection* consists of using RMB on it.

To *add to the selection*, use ⇧ Shift RMB on more objects.

If the *objects are overlapping* in the view, you can use Alt RMB to cycle through possible choices.

If you want *to add to a selection* this way then the shortcut becomes ⇧ ShiftAlt RMB.

To *activate an object* that is already selected, click ⇧ Shift RMB on it.

To *deselect* an active object, click ⇧ Shift RMB one time - and hence two clicks if the object isn’t active. Note that this only works if there are no other objects under the mouse. Otherwise it just adds those to the selection. There appears to be no workaround for this bug.

Rectangular or Border Select

Mode: Object mode and Edit mode

Hotkey: B

Menu: Select → Border Select

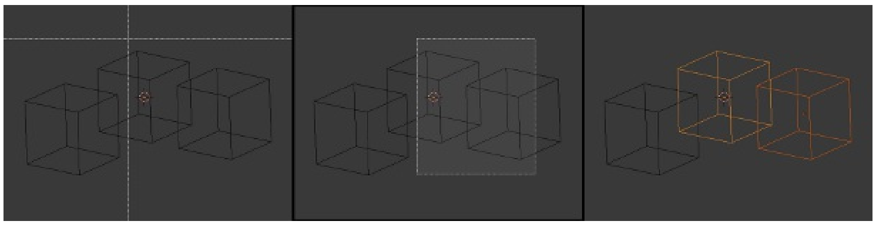
Description

With Border Select you draw a rectangle while holding down LMB. Any object that lies even partially within this rectangle becomes selected.

For deselecting objects, use MMB or Border Select again with holding ⇧ Shift.

To cancel the selection use RMB.

Example


Border selecting in three steps

Border Select has been activated in the first image and is indicated by showing a dotted cross-hair cursor. In the second image, the *selection region* is being chosen by drawing a rectangle with the LMB. The rectangle is only covering two cubes. Finally, in the third image, the selection is completed by releasing LMB.

Notice in the third image, the bright color of left-most selected cube. This means it is the “active object”, the last selected object prior to using the Border Select tool.

Hints

Border Select adds to the previous selection, so in order to select only the contents of the rectangle, deselect all with A first.

Lasso Select

Mode: Object mode and Edit mode

Hotkey: Ctrl LMB

Menu: no entry in the menu

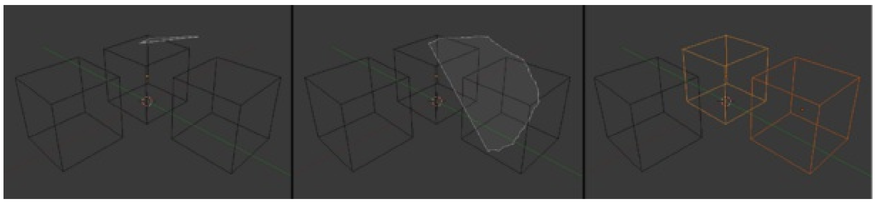
Description

Lasso select is used by drawing a dotted line around the pivot point of the objects, in Object mode.

Usage

While holding Ctrl down, you simply have to draw around the pivot point of each object you want to select with LMB.

Lasso select adds to the previous selection. For deselection, use Ctrl⇧ Shift LMB.


Lasso selection example

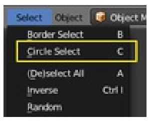
Circle Select

Mode: Object mode and Edit mode

Hotkey: C

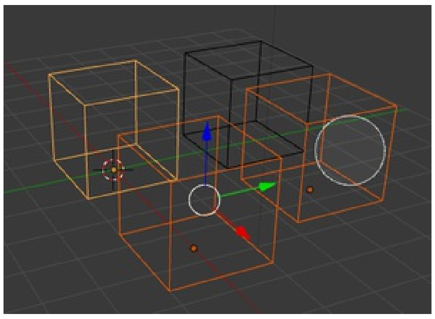
Menu: Select → Circle Select

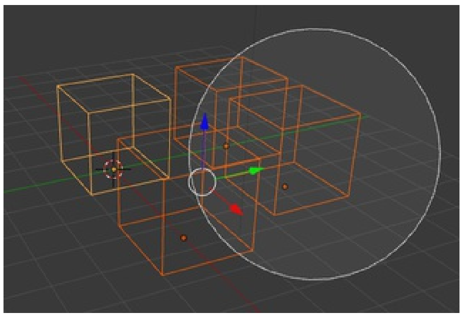
Description



Main selection menu

Circle Select is used by moving with dotted circle through objects with LMB. You can select any object by touching of circle area. It is possible to dynamically change the diameter of circle by scrolling MMB as seen in pictures below. Deselection is under the same principle - MMB. To cancel the selection use RMB or key Esc,


Circle selection


...with huge circle

Menu Selection

The selection methods described above are the most common. There are also many more options accessible through the Select menu of the 3D view.

Each is more adapted to certain operations.

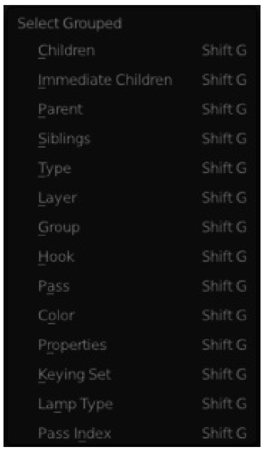
Select Grouped

Mode: Object mode

Hotkey: ⇧ ShiftG

Menu: Select → Grouped

Description



Select Grouped menu

There are two ways to organize the objects in relation to one another. The first one is parenting, and the second is simple grouping. We can use these relationships to our advantage by selecting members of respective families or groups.

Options

Select → Grouped in Object mode uses the active object as a basis to select all others.

Available options are:

Children
: Selects all children of the active object recursively.

Immediate Children
: Selects all direct children of the active object.

Parent
: Selects the parent of this object if it has one.

Siblings
: Select objects that have the same parent as the active object. This can also be used to select all root level objects (objects with no parents).

Type
: Select objects that are the same type as the active one.

Layer
: Objects that have at least one shared layer.

Group
: Objects that are part of a group (rendered green with the default theme) will be selected if they are in one of the groups that the active object is in.

Object Hooks
: Every hook that belongs to the active object.

Pass
: Select objects assigned to the same render pass. Render passes are set in Properties → Object → Relations and can be used in the Node Compositor (Add → Convertor → ID Mask.)

Color
: Select objects with same Object Color. Object colors are set in Properties → Object → Display → Object Color.)

Properties
: Select objects with same Game Engine Properties.

Keying Set
: Select objects included in active Keying Set.

Lamp Type
: Select matching lamp types.

Pass Index
: Select matching object pass index.

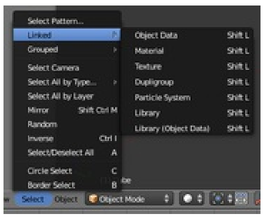
Select linked

Mode: Object mode

Hotkey: ⇧ ShiftL

Menu: Select → Linked

Description



Linked selection menu

Selects all objects which share a common datablock with the active object.

Options

Select → Linked in Object mode uses the active object as a basis to select all others.

Available options are:

Object Data
: Selects every object that is linked to the same Object Data, i.e. the datablock that specifies the type (mesh, curve, etc.) and the build (constitutive elements like vertices, control vertices, and where they are in space) of the object.

Material
: Selects every object that is linked to the same material datablock.

Texture
: Selects every object that is linked to the same texture datablock.

Dupligroup
: Selects all objects that use the same Group for duplication.

Particle System
: Selects all objects that use the same Particle System

Library
: Selects all objects that are in the same Library

Library (Object Data)

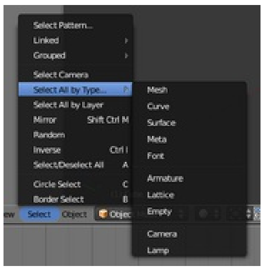
Select All by Type

Mode: Object mode

Hotkey: None

Menu: Select → Select All by Type

Description



By Type selection menu

The types are Mesh, Curve, Surface, Meta, Font, Armature, Lattice, Empty, Camera, Lamp, Speaker.

With this tool it becomes possible to select every **visible** object of a certain type in one go.

Options

Select All by Type in Object mode offers an option for every type of object that can be described by the ObData datablock.

Just take your pick.

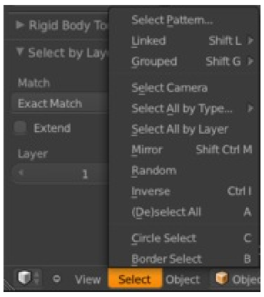
Select All by Layer

Mode: Object mode

Hotkey: None

Menu: Select → Select All by Layer

Description



All by Layer selection menu

Layers are another means to regroup your objects to suit your purpose.

This option allows the selection of every single object that belongs to a given layer, visible or not, in one single command.

Options

In the Tool Shelf → Select by Layer the following options are available:

Match
: The match type for selection.

Extend
: Enable to add objects to current selection rather than replacing the current selection.

Layer
: The layer on which the objects are.

Selection of Objects

Rather than using the Select All by Layer option, it might be more efficient to make the needed layers visible and use A on them. This method also allows objects to be deselected.

Other Menu Options

Available options on the first level of the menu are:

Select Pattern...
: Selects all objects whose name matches a given pattern. Supported wildcards: * matches everything, ? matches any single character, [abc] matches characters in "abc", and [!abc] match any character not in "abc". The matching can be chosen to be case sensitive or not.
As an example *house* matches any name that contains "house", while floor* matches any name starting with "floor".

Select Camera
: Select the active camera.

Mirror (⇧ ShiftCtrlM)
: Select the Mirror objects of the selected object eg. L.sword -> R.sword.

Random
: Randomly selects unselected objects based on percentage probability on currently active layers. On selecting the command a numerical selection box becomes available in the Tool Shelf.
It’s important to note that the percentage represents the likelihood of an unselected object being selected and not the percentage amount of objects that will be selected.

Inverse (CtrlI)
: Selects all objects that were not selected while deselecting all those which were.

(De)select All (A)
: If anything was selected it is first deselected. Otherwise it toggles between selecting and deselecting every visible object.

Introduction

In this section will be described tools for editing objects in Object Mode.

Information about some additional possibilities are described in Manipulation in 3D.

Object Mode

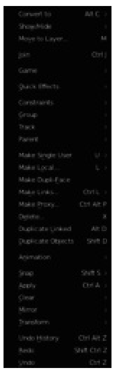
Object Mode button

By default new files opens with enabled Object Mode. To enable it you may in 3D View window → Header click Object Mode button (see picture Object Mode button)

All edition tools works only with selected objects. See Selecting Objects for more information.

All commands described below can be found in *Object* menu and/or in *Object tools* panel (see pictures).

Object tools panel



Object menu

Creation and deletion

The most basic edition includes manipulation with existence of objects. Below are listed different types of creation and deletion tools.

Add

Hotkey: ⇧ ShiftA

Menu: Main → Add

We may add one of those primitives:

- **Mesh**: Plane, Cube, Circle, UV Sphere, Icosphere, Cylinder, Cone, Grid, Monkey, Torus.
- **Curve**: Bezier, Circle, NURBS Curve, NURBS Circle, Path.
- **Surface**: NURBS Curve, NURBS Circle, NURBS Surface, NURBS Cylinder, NURBS Sphere, NURBS Torus.
- **Metaball**: Ball, Capsule, Plane, Ellipsoid, Cube.
- **Text**.
- **Armature**: Single Bone.
- **Lattice**.
- **Empty**: Plane Axis, Arrows, Single Arrow, Circle, Cube, Sphere, Cone, Image.
- **Speaker**.
- **Camera**.
- **Lamp**: Point, Sun, Spot, Hemi, Area.
- **Force Field**: Force, Wind, Vortex, Magnetic, Harmonic, Charge, Lennard-Jones, Texture, Curve Guide, Boid, Turbulence, Drag, Smoke Flow.
- **Group Instance**: (user defined groups of objects).

Duplicate

Hotkey: ⇧ ShiftD

Menu: *Object → Duplicate Objects*

Hotkey: AltD

Menu: *Object → Duplicate Linked*

Duplication makes exact copy of objects. May be linkage of some attributes depending on specific tool. See Duplication for more information.

Join

Hotkey: CtrlJ

Menu: *Object → Join*

Joining makes one single object from all selected objects. Objects must be of the same type. Origin point is obtained from the previously *active* object. Performing a join is equivalent to adding new objects while in Edit mode. The non-active objects are deleted (their meshes were taken by active object). Only the active object remains. This only works with editable objects, containing meshes and curves.

Delete

Hotkey: X, D or Delete, D

Menu: *Object → Delete... → Delete*

Deletion erases selected objects.

Transformation tools

Objects can be transformed in a variety of ways. Below are listed different types of transformation.

Translate

Hotkey: G

Menu: *Object → Transform → Grab/Move*

Translation means changing location of objects. This changes X, Y and/or Z coordinates of object's Origin point relative to center of coordinates.

Rotate

Hotkey: R

Menu: *Object → Transform → Rotate*

Rotation means changing angles of objects. This changes rotation angles around X, Y and/or Z axes of object's coordinate system relative to current coordinate system. No parts of each object are changing their position relative to other parts of the same object.

Scale

Hotkey: S

Menu: *Object → Transform → Scale*

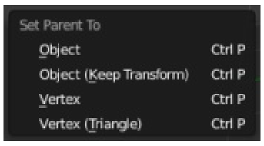
Scaling means changing proportions of objects. This proportionally stretches object along X, Y and/or Z axes of object's coordinate system.

Grouping And Parenting Objects

There can be many objects in a scene: A typical stage scene consists of furniture, props, lights, and backdrops. Blender helps you keep everything organized by allowing you to group like objects together.

When modeling a complex object, such as a watch, you may choose to model the different parts as separate objects. However, all of the parts may be attached to each other. In these cases, you want to designate one object as the parent of all the children. Movement, rotation or scaling of the parent also affects the children.

Parenting objects



Set Parent To pop-up menu

To parent objects, select at least two objects (select the *Child Objects* first, and select the *Parent Object* last), and press CtrlP. The Set Parent To dialog will pop up allowing you to select from one of several possible different parenting types. Selecting one of the entries in Set Parent To confirms, and the child/children to parent relationship is created.

The last object selected will be the *Active Object* (outlined in light orange), and will also be the *Parent Object*. If you selected multiple objects before selecting the parent, they will all be children of the parent and will be at the same level of the hierarchy (they are "siblings").

The *Set Parent To* popup dialog is context sensitive, which means the number of entries it displays can change depending on what objects are selected when the CtrlP shortcut is used.

For non-inverse-mode, press ⇧ ShiftCtrlP instead. This creates an alternative parent-child-relationship where child-objects exist entirely in the parent's coordinate system. This is the better choice for CAD purposes, for example.

Moving, rotating or scaling the parent will also usually move/rotate/scale the child/children. However moving/rotating/scaling the child/children of the parent will not result in the parent moving/rotating/scaling. In other words, the direction of influence is from parent to child and not child to parent.

In general when using the CtrlP or [3D View Editor Header > Object Menu > Parent Menu] entries to parent objects, the *Child Objects* can only have one *Parent Object*. If a *Child Object* already has a *Parent Object* and you give it another parent then Blender will automatically remove the previous parent relationship.

Blender supports many different types of parenting, listed below:

- Object
- Object (Keep Transform)
- Armature Deform > With Empty Groups
- Armature Deform > With Automatic Weights
- Armature Deform > With Envelope Weights
- Bone
- Bone Relative
- Curve Deform
- Follow Path
- Path Constraint
- Lattice Deform
- Vertex
- Vertex (Triangle)
- Clear Parent
- Clear And Keep Transformation
- Clear Parent Inverse

Object Parent

Object Parent is the most general form of parenting that Blender supports. If will take selected objects and make the last selected object the *Parent Object*, while all other selected objects will be *Child Objects*.

Object (Keep Transform) Parent

Object (Keep Transform) Parent works in a very similar way to *Object Parent* the major difference is in whether the *Child Objects* will remember any previous transformations applied to them from the previous *Parent Object*.

Since explaining this in an easy to understand technical way is hard, lets instead use an example to demonstrate.

Assume that we have a scene consisting of 3 objects, those being 2 Empty Objects named "EmptyA" and "EmptyB", and a Monkey object. See figure 1.

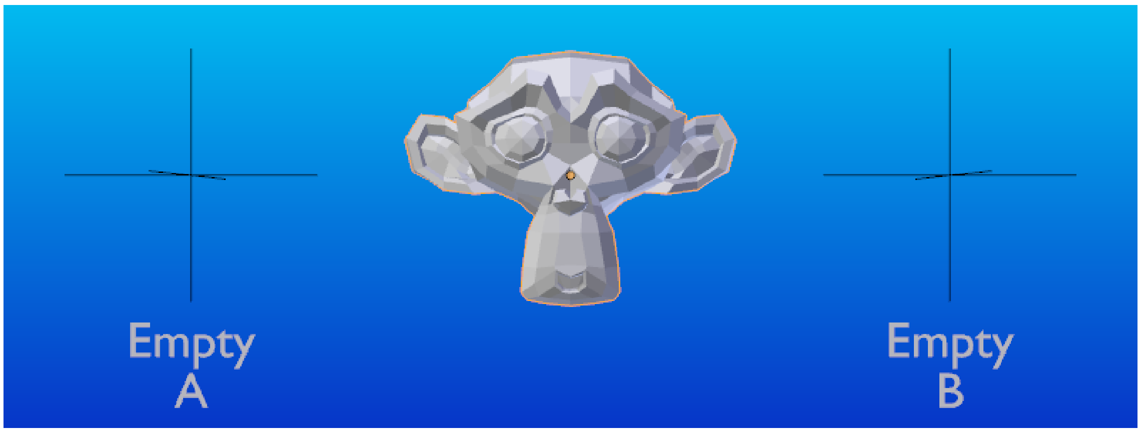


Figure 1 - Scene with 2 Empties and a Monkey, no parenting currently active.

Figure 1 shows the 3 objects with no parenting relationships active on them.

If you select the Monkey object by RMB click and then ⇧ Shift RMB click "EmptyA" object and press CtrlP and then select *Object* from the *Set Parent To* popup dialog box. This will result in "EmptyA" object being the *Parent Object* of the Monkey object. With only "EmptyA" selected rotating/scaling/moving it will result in the Monkey object being altered respectively.

Scale the "EmptyA" object, so that the Monkey becomes smaller and moves to the left a little. See figure 2.

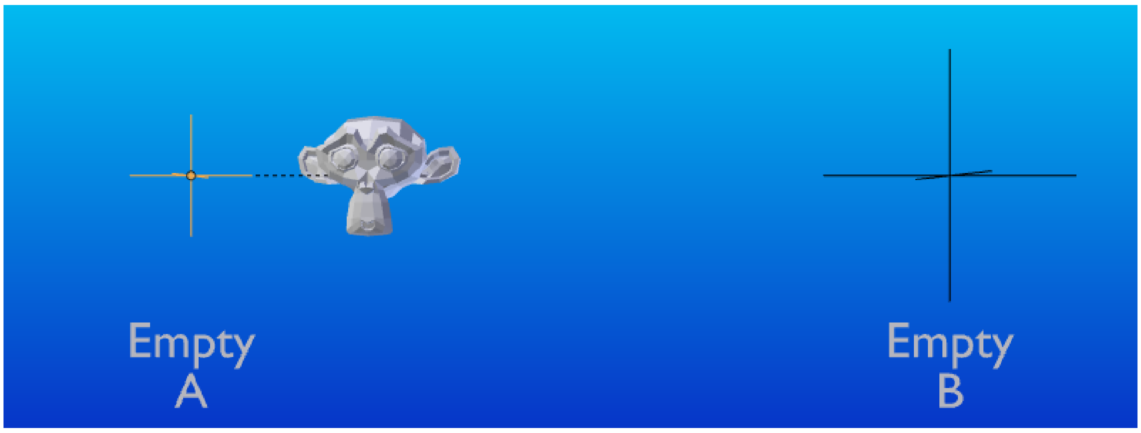


Figure 2 - Scene with Monkey object being the *Child Object* of "EmptyA". "EmptyA" has been scaled resulting in the Monkey also being scaled and moved to the left.

If you select only the Monkey object by RMB click and then ⇧ Shift RMB click "EmptyB" object and press CtrlP and select *Object* from the *Set Parent To* popup dialog box. This will result in "EmptyB" object being the *Parent Object* of the Monkey object. Notice that when you change the parent of the Monkey the scale of the Monkey changed. See figure 3.

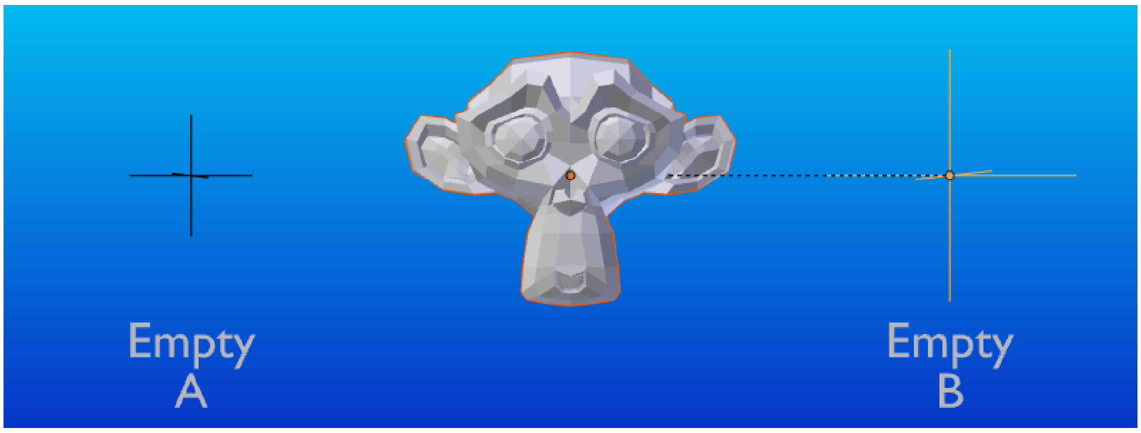


Figure 3 - Scene with Monkey object having its a parent changed from "EmptyA" to "EmptyB" and the resulting change in scale.

This happens because the Monkey object never had its scale altered directly, the change came about because it was the child of "EmptyA" which had its scale altered. Changing the Monkey's parent to "EmptyB" resulted in those indirect changes in scale being removed, because "EmptyB" has not had its scale altered.

This is often the required behaviour, but it is also sometimes useful that if you change your *Parent Object* that the *Child Object* keep any previous transformations it got from the old *Parent Object*; If instead when changing the *Parent Object* of the Monkey from "EmptyA" to "EmptyB" we had chosen parenting type *Object (Keep Transform)*, the Monkey would keep its scale information it obtained from the old parent "EmptyA" when it is assigned to the new parent "EmptyB"; See Figure 4.

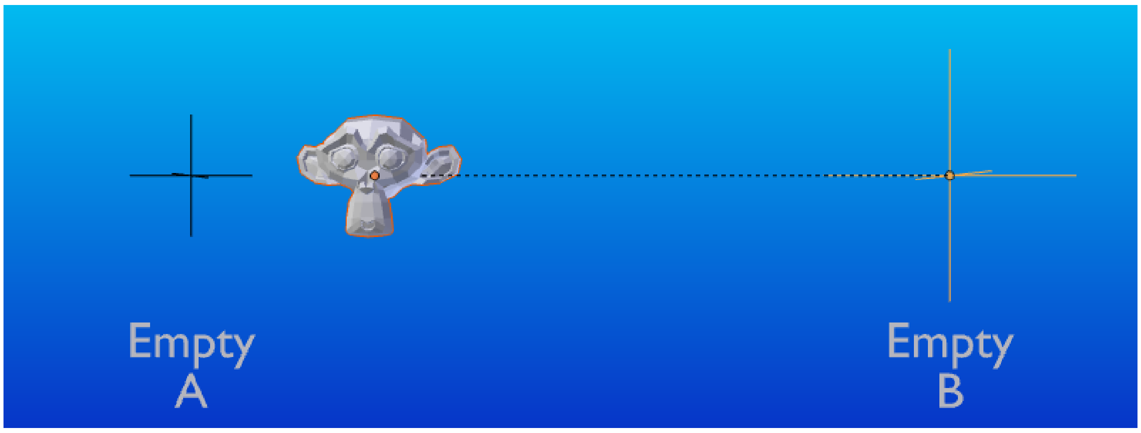


Figure 4 - Scene with Monkey object having its a parent changed from "EmptyA" to "EmptyB" using 'Object (Keep Transform)' parent method.

If you want to follow along with the above description here is the blend file used to describe *Object (Keep Transform)* parenting method:

File:Parent - Object (Keep Transform) (Demo File).blend

Armature Deform Parent

An Armature in Blender can be thought of as similar to the armature of a real skeleton, and just like a real skeleton an Armature can consist of many bones. These bones can be moved around and anything that they are attached to or associated with will move and deform in a similar way.

In Blender Armature Object Types are usually used to associate certain bones of an Armature to certain parts of a Mesh Object Types Mesh Geometry. You are then able to move the Armature Bones and the Mesh Object will deform. See figure 5.

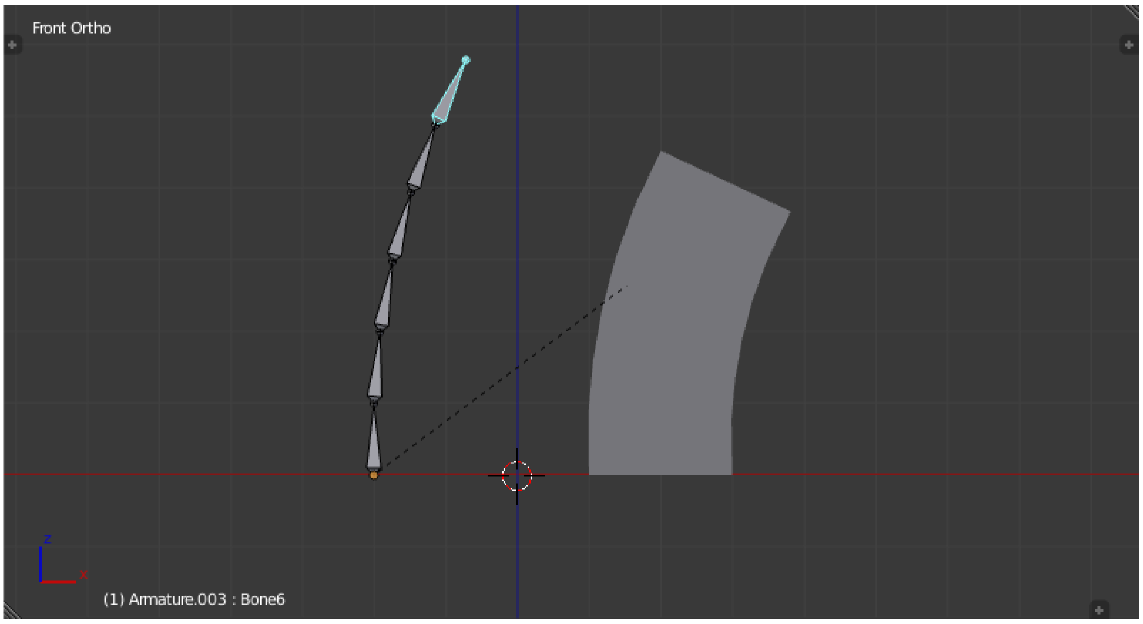


Figure 5 - Armature Object Bone associated with another Mesh Object, as the bone move the Mesh deforms similarly.

Armature Deform Parenting is one of the most flexible ways of associating Bones in an Armature to another Object, it gives a lot of freedom but that comes at the price of a little complexity, as there are multiple steps involved in setting up Armature Deform Parenting such that deformations are actually carried out.

Blender has several different ways of Parenting an Armature to an object, most of them can automate several of the steps involved, but all of them ultimately do all the steps we describe for Armature Deform Parenting.

Using the Armature Deform Parenting operator is the first step in setting up the relationship between an Armature Object and its Child Objects.

To use Armature Deform Parenting you must first select all the Child Objects that will be influenced by the Armature and then lastly select the Armature Object itself. Once all the Child Objects and the Armature Object are selected press CtrlP and select Armature Deform in the Set Parent To popup dialog. See figure 6.

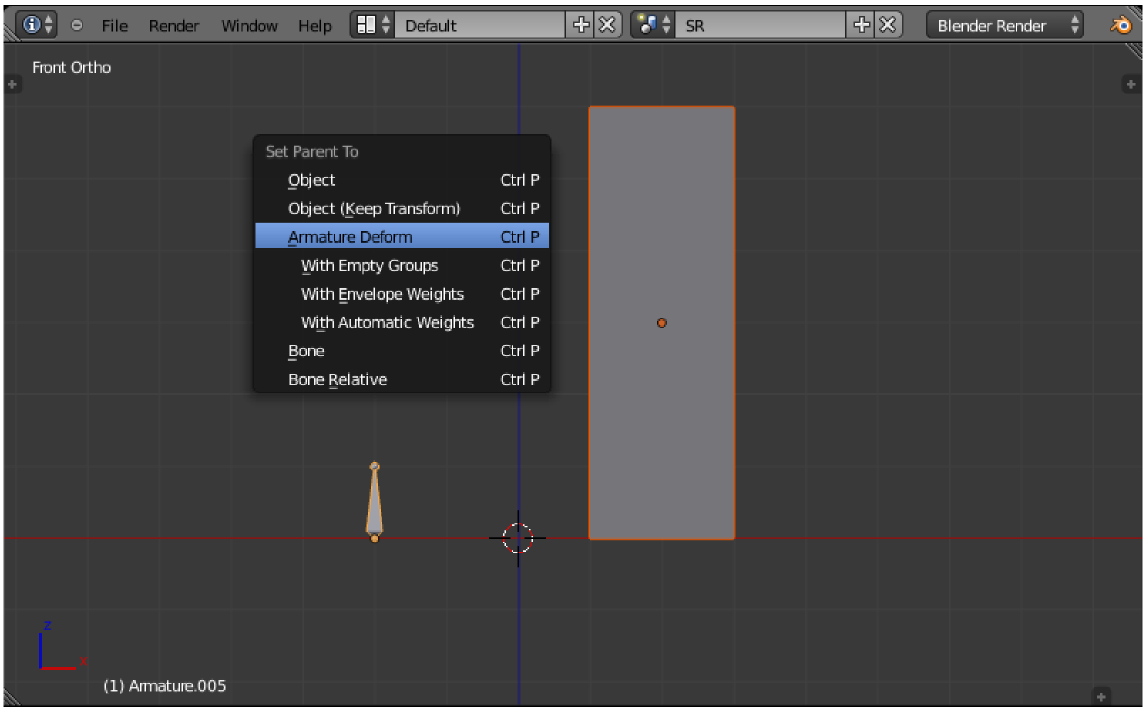


Figure 6 - Set Parent To dialog with Armature Deform Parenting option highlighted.

Once this is done the Armature Object will be the Parent Object of all the other Child Objects, also we have informed Blender that the Bones of the Armature Object can be associated with specific parts of the Child Objects so that they can be directly manipulated by the Bones.

At this point however all Blender knows is that the Bones of the Armature could be used to alter the Child Objects, we haven't yet told Blender which Bones can alter which Child Objects or by how much.

To do that we must individually select each Child Object individually and toggle into Edit Mode on that Child Object. Once in Edit Mode we can then select the vertices we want to be influenced by the Bones in the Armature. Then with the vertices still selected navigate to [Properties Editor > Object Data Context > Vertex Groups Panel] and create a new Vertex Group with the same name as the Bone that you want the selected vertices to be influenced by.

Once the Vertex Group has been created we then assign the selected vertices to the Vertex Group by clicking the Assign Button. By default when selected vertices are assigned to a Vertex Group they will have an Influence Weight of 1. This means that they are fully influenced when a Bone they are associated with is moved, if the Influence Weight had been .5 then when the bone moves the vertices would only move half as much. See figure 7.

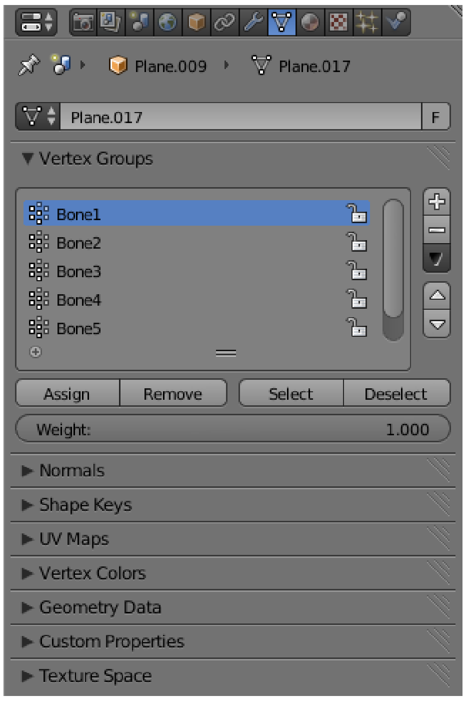


Figure 7 - Properties Editor > Object Data Context > Vertex Groups Panel with Assign Button and influence Weight Slider highlighted.

Once all these steps have been carried out, the Bones of the Armature Object should be associated with the Vertex Groups with the same names as the Bones. You can then select the Armature Object and switch to Pose Mode in the [3D View Editor Header > Mode Select Button]. See figure 8.

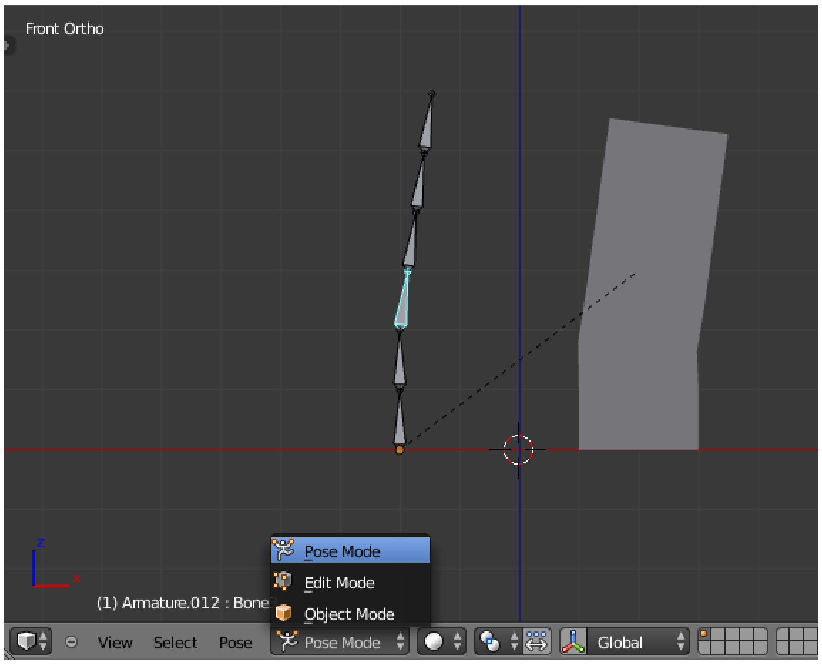


Figure 8 - 3D View Editor Header > Mode Select Button] set to Pose Mode, with Armature Bone highlighted in Cyan and effecting the Mesh Object

Once in Pose Mode transforming one of the Bones of the Armature that has been associated with vertices of an object will result in those vertices also being transformed.

Armature Deform Parenting - Example Of Use

What follows is a simple example of how to setup Armature Deform Parenting so that you end up with an Armature whose Bones can Influence the mesh of a Child Object when the Armature is in Pose Mode.

1. Open Blender with it's Default Scene.
2. Switch the 3D Viewport to Front View by pressing Numpad1.
3. Select the Default Cube and delete it by pressing the X.
4. Add a Plane Object by pressing the Spacebar and entering "add plane" into search text field. Then press the ↵ Enter or LMB the listed Add Plane entry.
5. While in Object Mode and with the Plane Object still selected we need to rotate it on x axis by 90 degrees, so that its face points towards the front view. To do this by pressing the R then X then enter the value 90.
6. Scale the Plane along the Z axis by a factor of 3. Do this by pressing the S then the Z then enter the value 3.
7. With the Plane still selected and still in Object Mode move it upward 3 Blender Units along the Z axis. Do this by pressing G then Z then enter the value 3.
8. Move the Plane along the X axis to the right by 2 Blender Units. Do this by pressing the G then X then enter the value 2.
9. If you are following along your scene should look similar to figure 9.

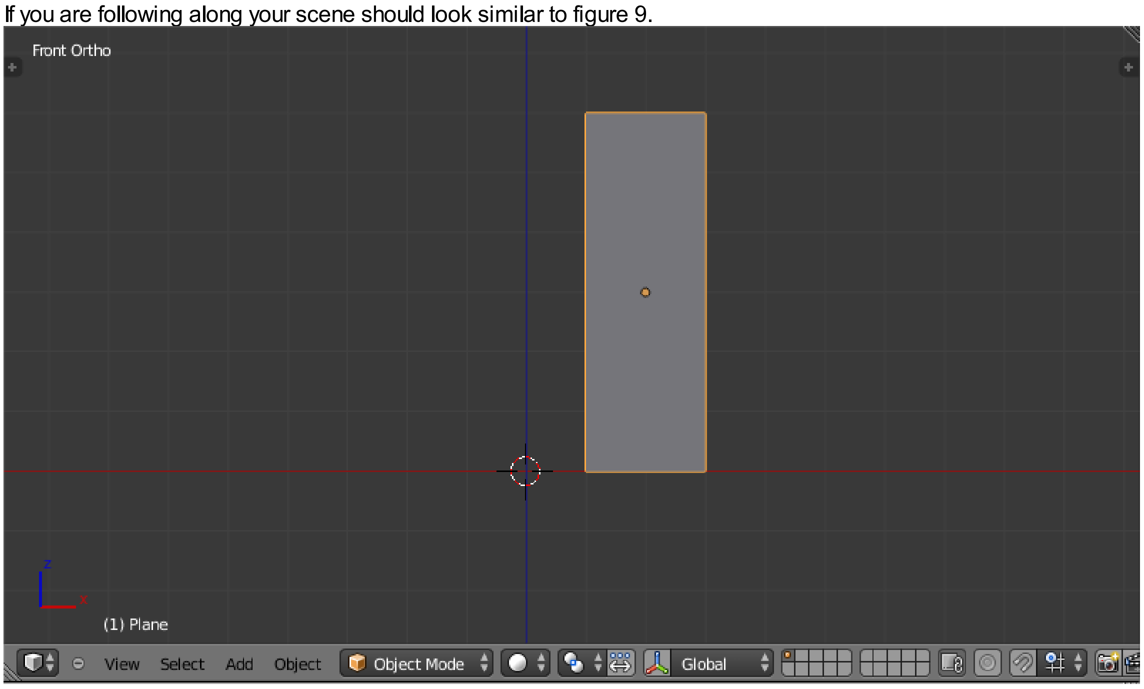
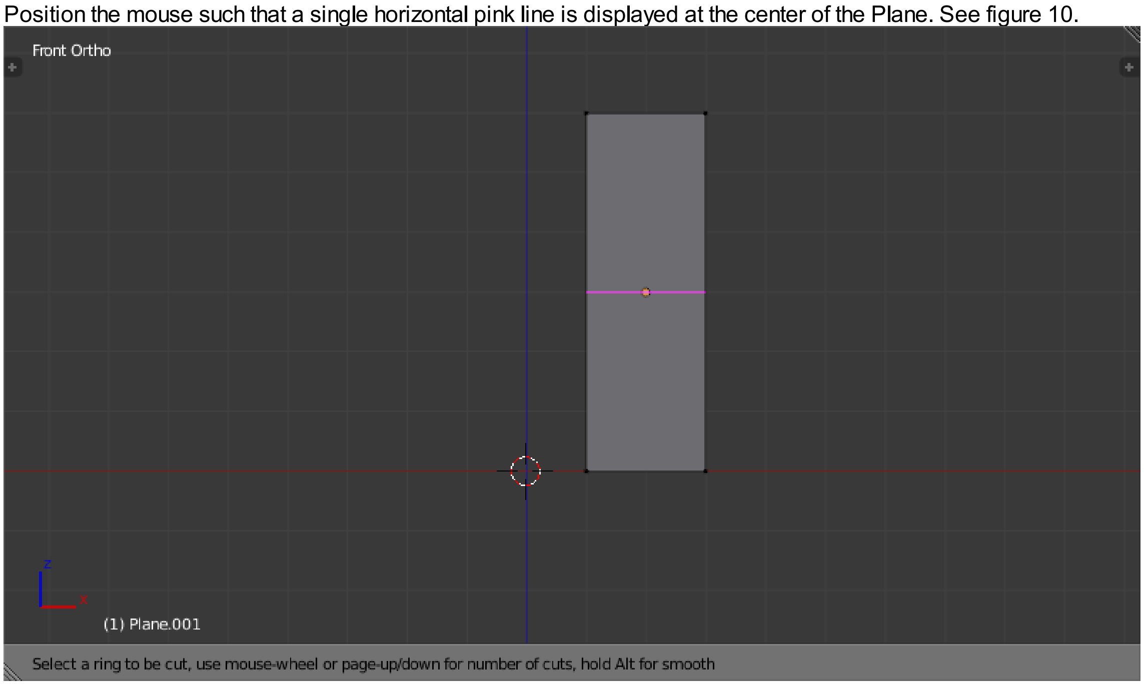


Figure 9 - 3D Viewport with Plane Positioned and Scaled in Front View.

10. With the Plane still selected toggle into Edit Mode by pressing ⇆ Tab until Edit Mode is displayed in the [3D View Editor Header > Mode Select Button].
11. Now make sure all the vertices of the Plane are selected by pressing the A repeatedly until all the vertices are selected (highlighted in orange).
12. With all the vertices selected we now need to tell Blender to split the face of the Plane up into 6 equal sized faces. We can do

this by using Blender's Loop Cut tool.

13. Active the Loop Cut tool by pressing CtrlR while the mouse is over the Plane object in the 3D Viewport.
14. Position the mouse over the middle section of the Plane. Depending on where you position the mouse over the Plane, there will be a thin pink line drawn either vertically or horizontally along the Plane.
15. Position the mouse such that a single horizontal pink line is displayed at the center of the Plane. See figure 10.



Select a ring to be cut, use mouse-wheel or page-up/down for number of cuts, hold Alt for smooth

Figure 10 - 3D Viewport with a Plane with Loop Cut active for a horizontal cut displaying single pink cut line in Front View.

16. With the pink line still displayed enter the value 5 then press the ↵ Enter twice, without moving the mouse.
17. If done correctly you should now have a Plane that is sectioned into 6 horizontal faces. See figure 11.

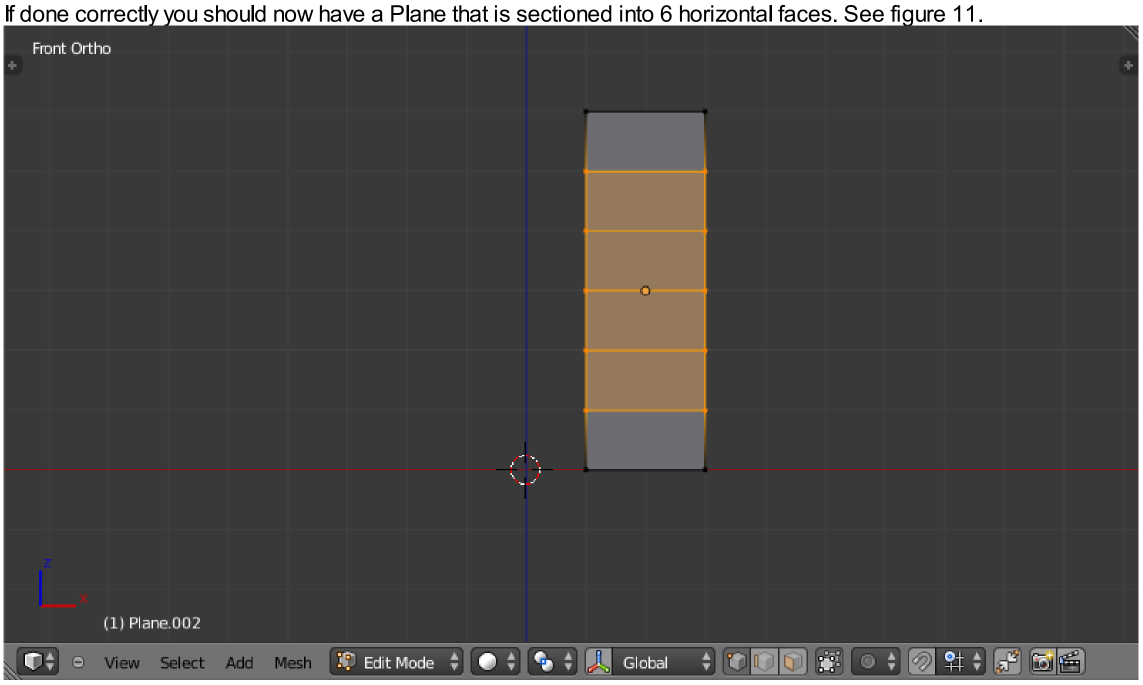


Figure 11 - 3D Viewport with a Plane sectioned into 6 faces using Loop Cut in Front View.

18. With the Plane still selected Toggle into Object Mode by pressing the ⇆ Tab repeatedly until Object Mode is displayed in the [3D View Editor Header].
19. In Object Mode add an Armature Object. Do this by positioning the mouse over the 3D Viewport then activating the Add Menu ⇧ ShiftA selecting the Armature entry and selecting the Single Bone entry.
20. Move the Armature to the left along the X axis by 2 Blender Units. Do this by selecting the Armature, pressing the G then X then enter the value -2.
21. If you were following along you should have a scene that looks similar to figure 12.

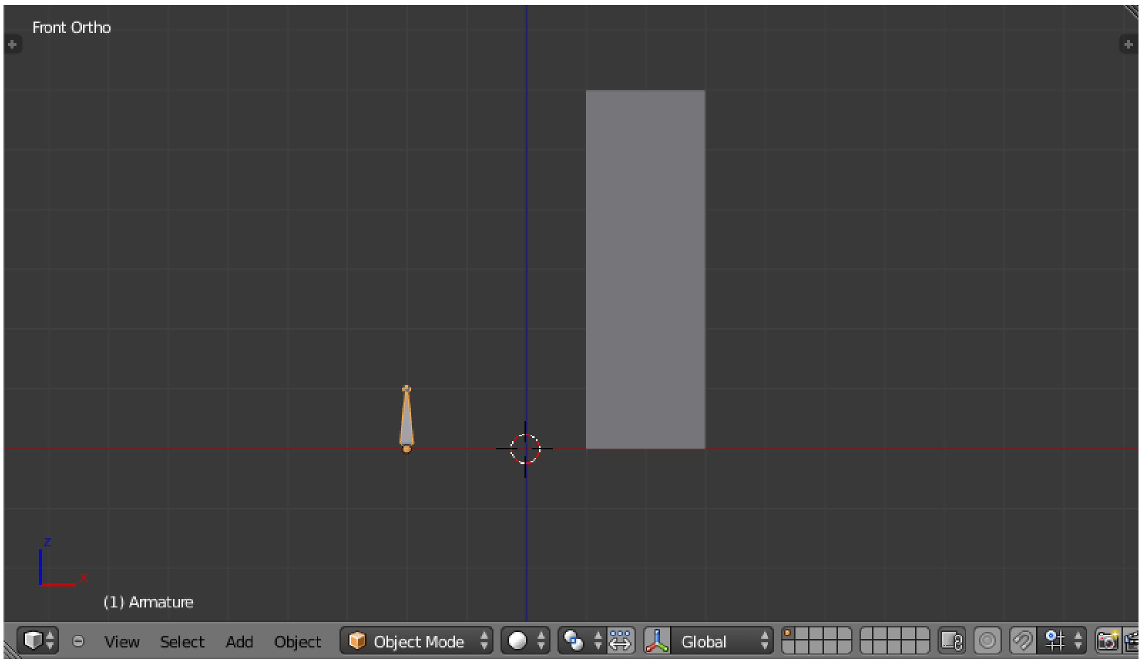


Figure 12 - 3D Viewport with the Armature Object and Plane positioned in Front View.

22. With the Armature still selected switch into Edit Mode by pressing the ⇆ Tab and toggling into Edit Mode.
23. The Armature Bone will display differently when it is in Edit Mode. See figure 13.

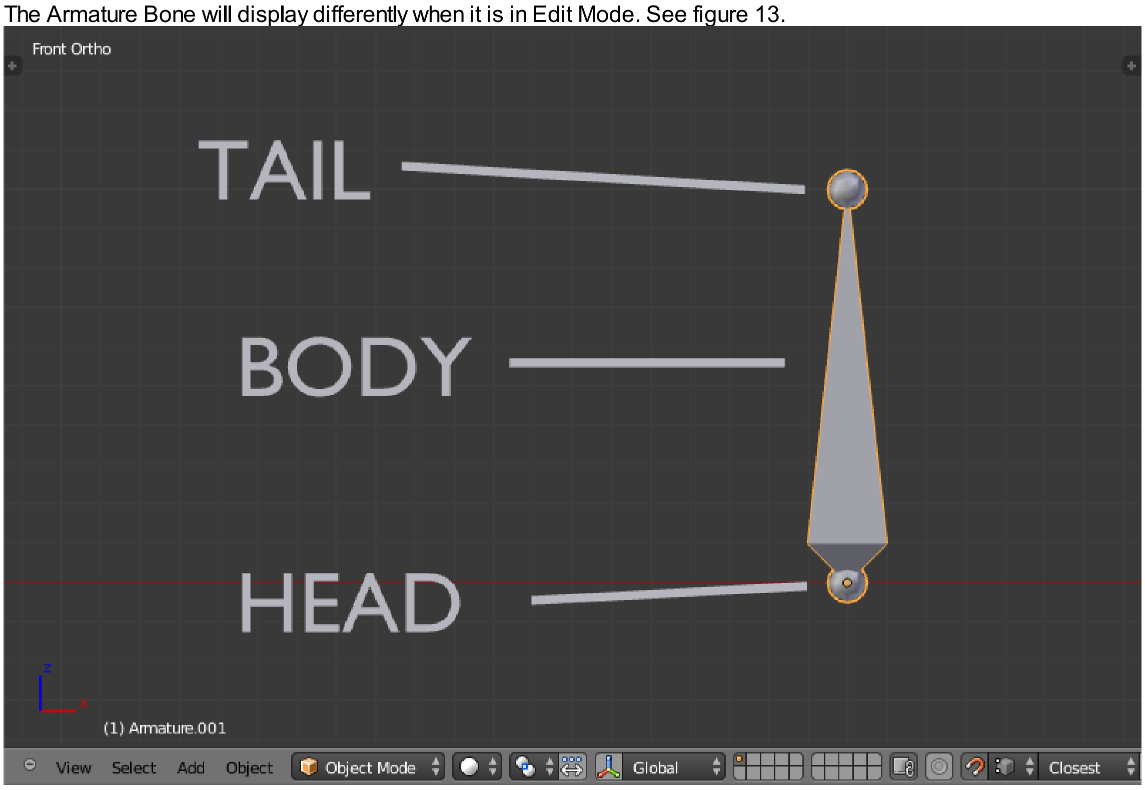


Figure 13 - 3D Viewport with Armature Bone in Edit Mode, with its Tail, Body and Head parts indicated in Front View.

24. The Armature Bone in Edit Mode has 3 distinct parts, The Head (thick bottom part), The Body (the middle part) and The Tail (the top pointy part).
25. Select The Tail of the Bone so that it is the only thing highlighted.
26. Extrude the Bone Tail by 1 Blender Unit along the Z axis. Do this by pressing the E then the Z and enter the value 1. This will extrude a new Bone from The Tail of the previous Bone.
27. Repeat the extruding from The Tail of the new bones 4 times by pressing ⇧ ShiftR 4 times. The ⇧ ShiftR executes the Repeat Last command operator.
28. You should now have an Armature with 6 bones in it, each emerging from The Tail of the previous bone.
29. If you have been following along you should have a scene that looks similar to figure 14.

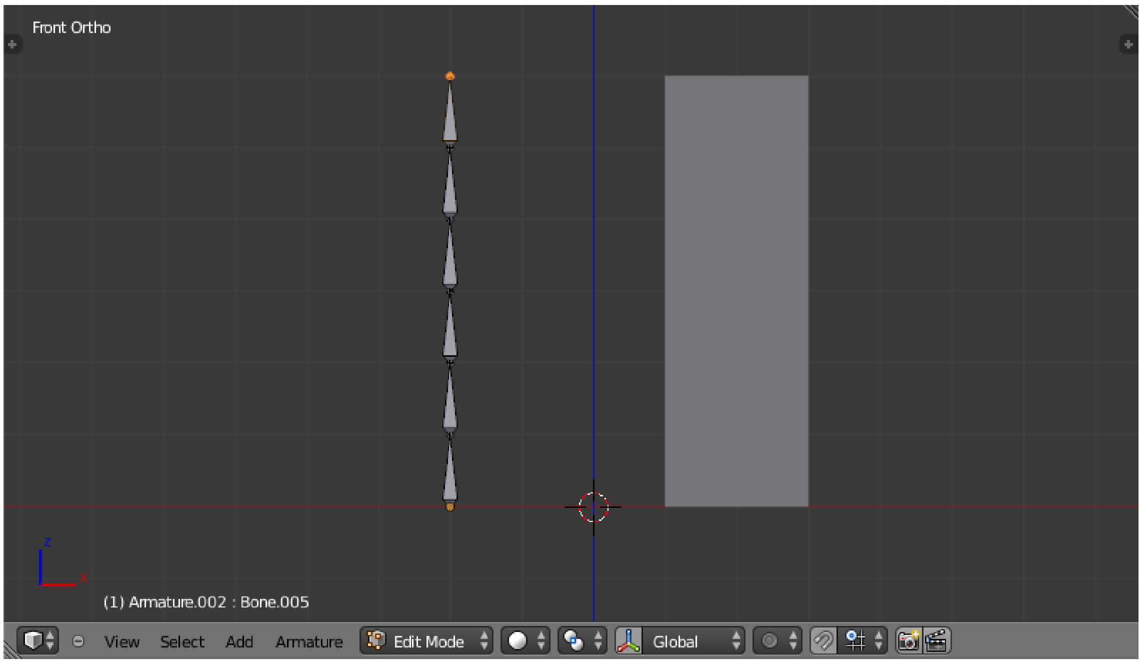

Front Ortho
(1) Armature.002 : Bone.005
View Select Add Armature Edit Mode Global

Figure 14 - Armature in Edit Mode, with 6 bones created and a Plane scaled and sectioned in to 6 pieces matching the length of the Armature in Front View.

30. While still in Edit Mode select the body of the first bone of the Armature we created. If selected correctly the entire bone will be highlighted, not just the head or tail.
31. Navigate to the [Properties Editor > Bone Context > Bone Text Field] and change the name of the bone from Bone to Bone1. See figure 15.

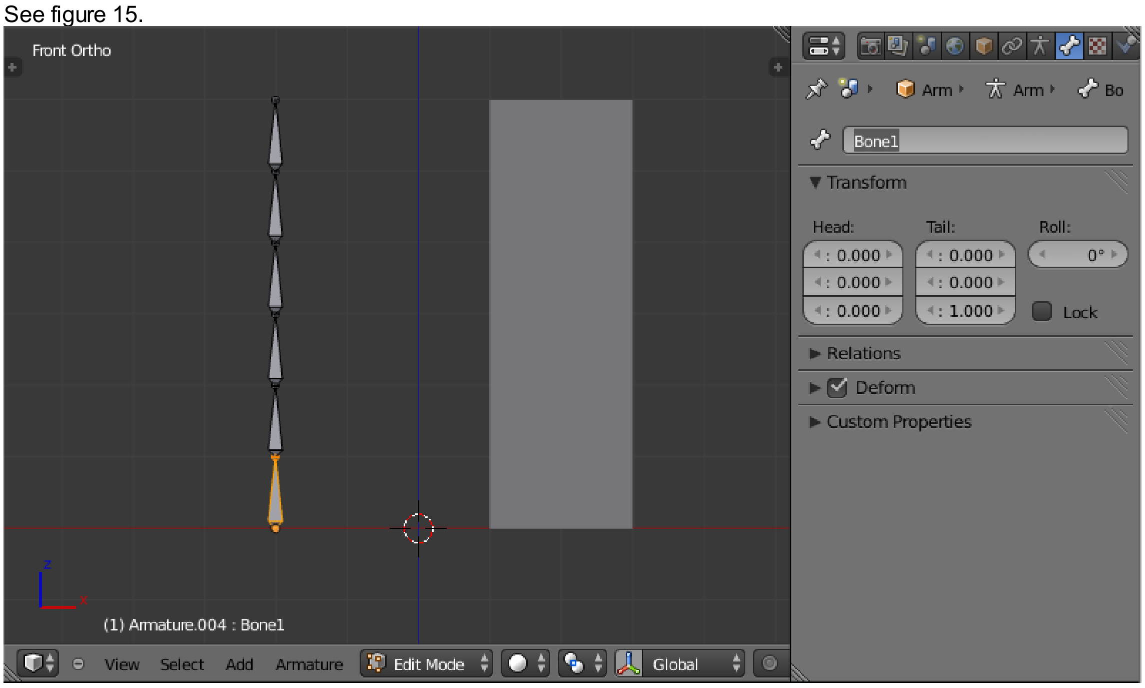


Figure 15 - The [Properties Editor > Bone Context > Bone Text Field] used to rename Bones of an Armature.

32. Do the same for all the other bones in the Armature naming them Bone2, Bone3, Bone4, Bone5 and Bone6 respectively.
33. Note that Blender is case sensitive when it comes to naming of objects such as bones, so make sure to name them consistently as Blender does not consider Bone or bone the same.
34. Once the bones have been renamed, toggle Blender into Object Mode by pressing the ⇆ Tab.
35. Now we need to make the Plane the Child Object of the Armature. To do this select only the Plane then ⇧ Shift RMB the Armature.
36. With the Plane and the Armature selected activate the Set Parent To popup dialog by pressing CtrlP and select the Armature Deform entry.
37. This should result in the Armature being the Parent Object of the Plane.
38. So at this point we have renamed all six bones of the Armature Bone1, Bone2, Bone3, Bone4, Bone5 and Bone6. We have split a single Plane into 6 faces and scaled it to match the total length of the Armature Bones. We then made the Plane the Child Object of the Armature using the CtrlP Armature Deform Parenting option.
39. The final step is to associate the different bones of the Armature to the different vertices that make up the faces of the Plane, so that moving the bones, actually moves the faces.
40. To do that we need to select the vertices we want to associate with a particular bone and then create a Vertex Group with the same name as the bone and then assign the selected vertices to that Vertex Group.
41. Make sure the Plane is the only object that is selected.
42. Switch into Edit Mode if we are not already by pressing the ⇆ Tab.
43. Switch to Face Selection Mode by pressing Ctrl⇆ Tab and select Face from the Mesh Select Mode popup dialog. See figure 16.

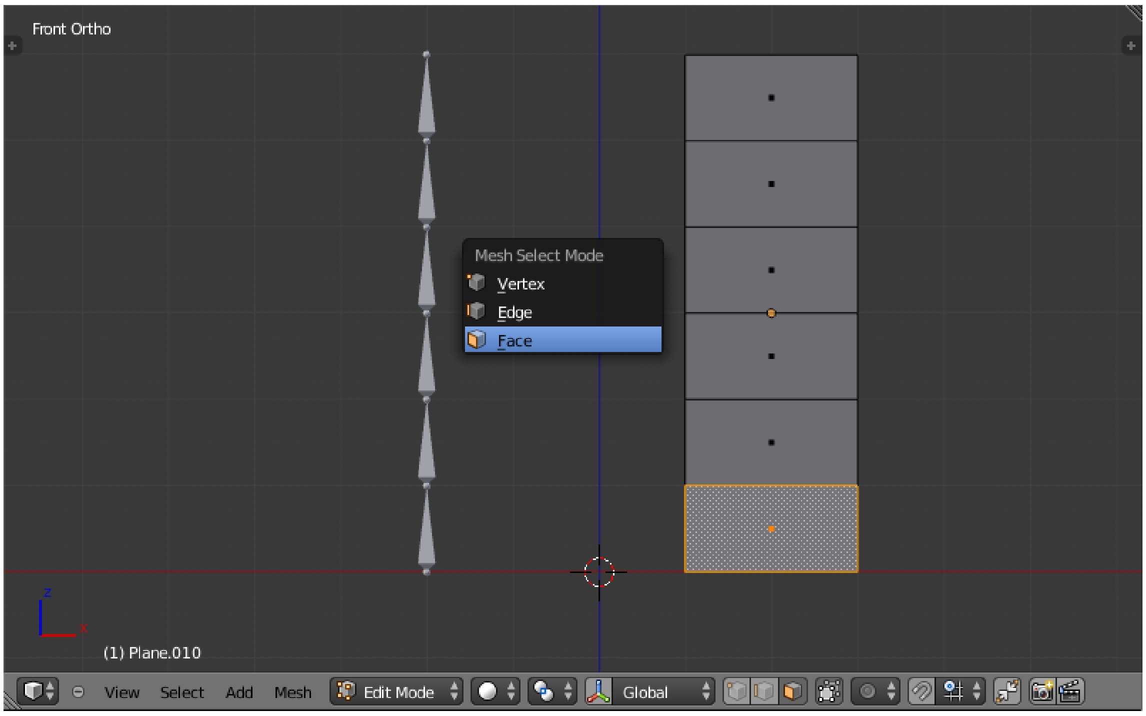


Figure 16 - Mesh Select Mode popup dialog with the Face entry selected.

44. Select the 1st face of the Plane by RMB it. See figure 17.

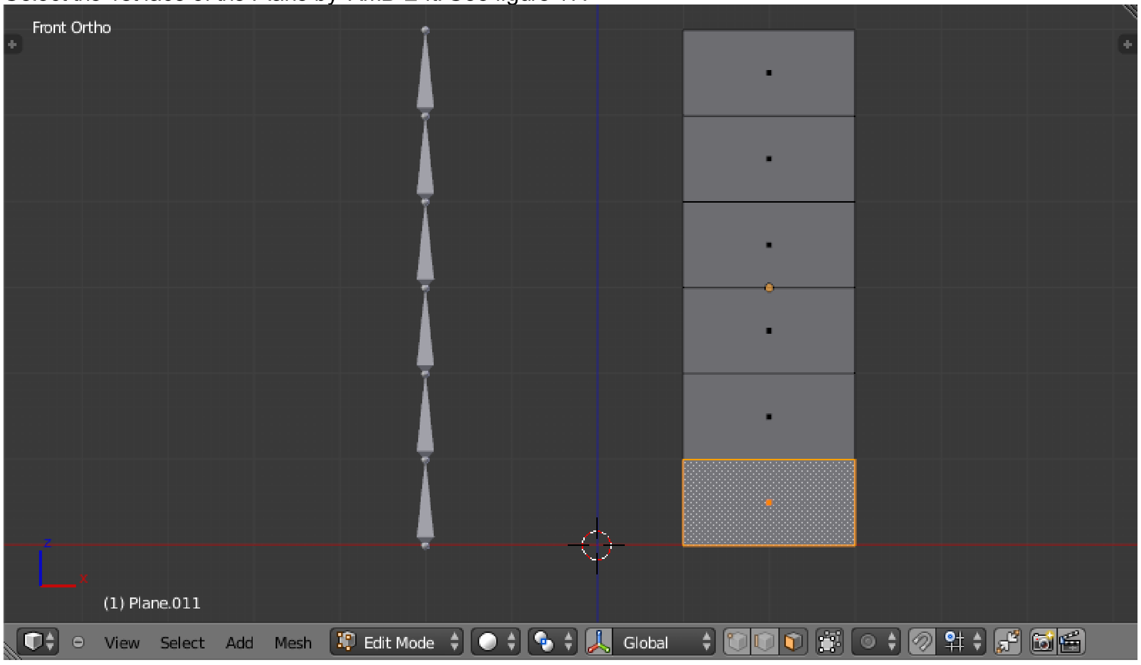


Figure 17 - Plane in Edit Mode with Face Select Mode active, with its 1st face selected.

45. Navigate to the [Properties Editor > Object Data > Vertex Groups Panel] and click the + button to the right of the Vertex Groups list. See figure 18.

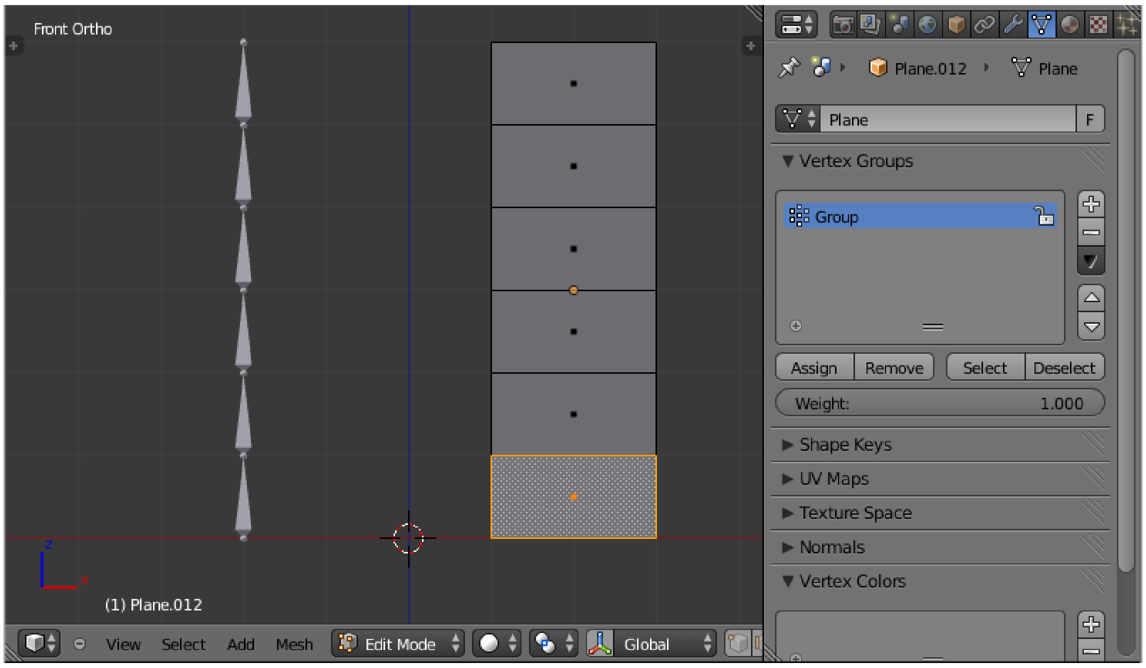


Figure 18 - [Properties Editor > Object Data Context > Vertex Groups Panel] highlighted and the + button highlighted.

46. This will create a new Vertex Group called Group.
47. Rename this newly created Group to Bone1 by Ctrl LMB on the Group name in the Vertex Group list and then typing in the new name. To complete the naming press the ↵ Enter.
48. Click the button named Assign to assign the currently selected face to that Vertex Group you just made and named Bone1.
49. At this point if all went well you will have associated the 1st face of the Plane with Bone1 of the Armature.
50. To associate the other faces 2 through 6 to the Armature Bones 2 through 6 you do the same things; Select the face, create a Vertex Group with the same name as the bone you want to associate the face with and then click the Assign button to associate the selected face with the selected Vertex Group.
51. Toggle into Object Mode by pressing the ⇆ Tab.
52. Now select only the Armature by RMB it.
53. Now we need to switch the Armature to Pose Mode. Do that by navigating to [3D View Editor Header > Mode Select Button] and LMB it and select the Pose Mode entry. See figure 19.

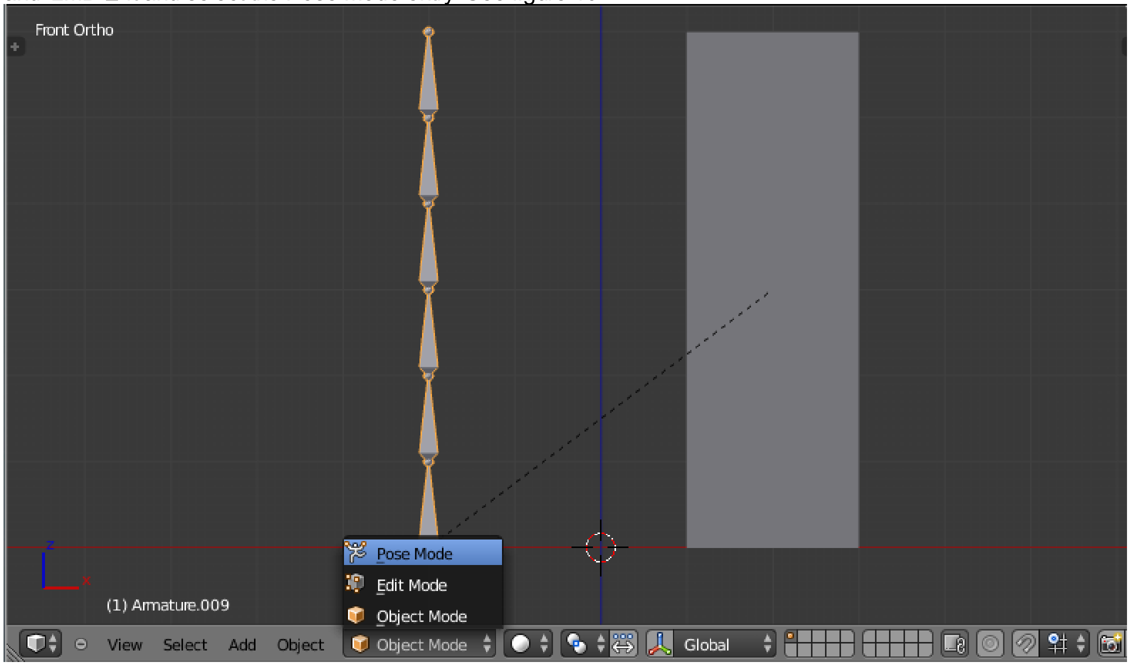


Figure 19 - [3D View Editor Header > Mode Select Button] with Pose Mode entry highlighted.

54. Now if you select the third bone (Bone3) in the Armature you will notice it gets highlighted in a Cyan color when in Pose Mode.
55. If you transform that bone in anyway the associated face on the Plane will also transform in a similar way.
56. For example rotate Bone3 and face section 3 will rotate in a similar way on the Plane. See figure 20.

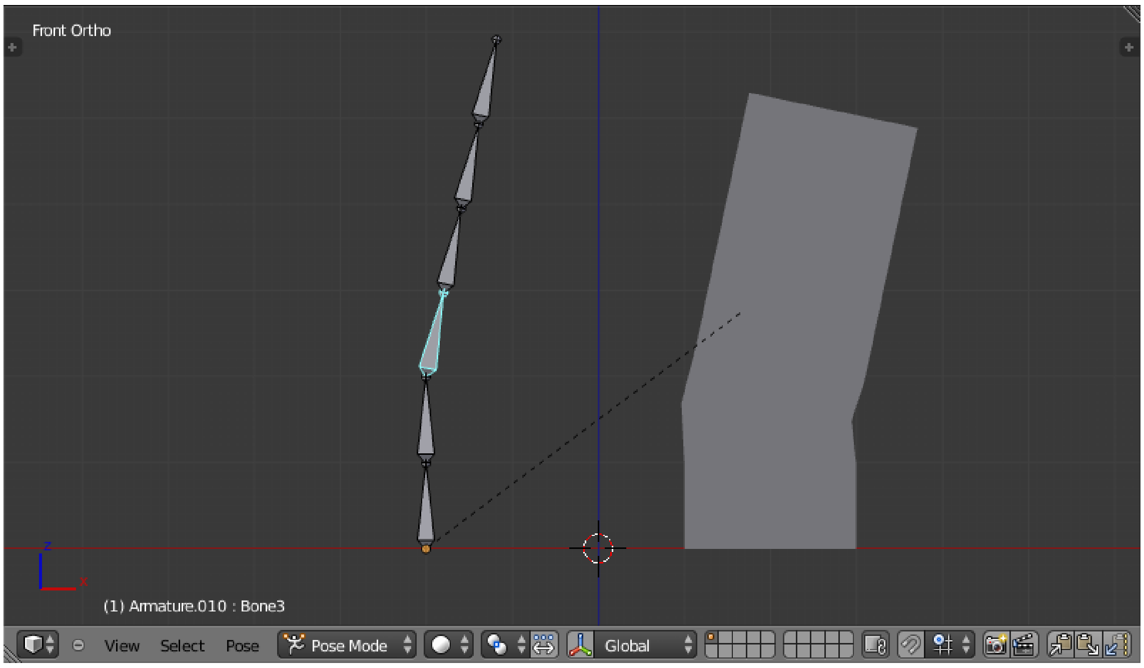


Figure 20 - The Armature in Pose Mode with Bone3 selected and rotated showing the effect on the Plane.

Armature Deform Parent With Empty Groups

The Armature Deform With Empty Groups parenting method works in almost the same way as Armature Deform parenting with one difference. That difference is that when you parent a Child Object to an Armature Object the names of the bones in the armature are copied to the Child Objects in the form of newly created Vertex Groups, one for each different deforming armature bone name. The newly created Vertex Groups will be empty this means they will not have any vertices assigned to those Vertex Groups. You still must manually select the vertices and assign them to a particular Vertex Group of your choosing to have bones in the armature influence them.

For example if you have an Armature Object which consists of 3 bones named BoneA, BoneB and BoneC and Cube Mesh Object type called Cube. If you parent the Cube Child Object to the Armature Parent Object the Cube will get 3 new Vertex Groups created on it called BoneA, BoneB and BoneC. Notice that each Vertex Group is empty. See figure 21.

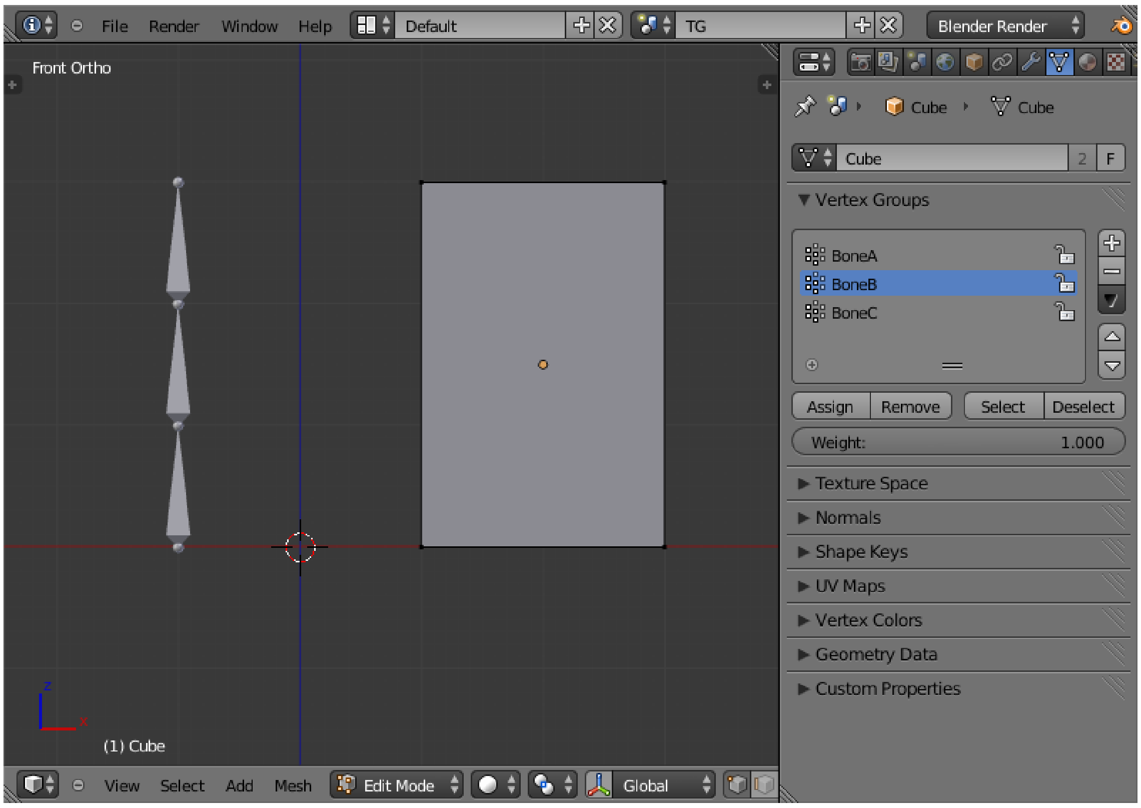


Figure 21 - Cube in Edit Mode showing the 3 created Vertex Groups after it was parented using Armature Deform With Empty Groups to an Armature with 3 Bones named BoneA, BoneB and BoneC with the Vertex Group Panel shown. All the Vertex Groups are empty.

Bones in an Armature can be generally classified into 2 different types:

- Deforming Bones
- Control Bones

Deforming Bones - Are bones which when transformed will result in vertices associated with them also transforming in a similar way. Deforming Bones are directly involved in altering the positions of vertices associated with their bones.

Control Bones - Are Bones which act in a similar way to switches, in that they control how other bones or objects react when they are transformed. A Control Bone could for example act as a sliding switch control, when the bone is in one position to the left it could indicate to other bones that they react in a particular way when transformed, when the Control Bone is positioned to the right, transforming other bones or objects could do something completely different. Control Bones are not directly used to alter the positions of vertices, in fact Control Bones often have no vertices directly associated with themselves.

When using the Armature Deform With Empty Groups parenting method Vertex Groups on the Child Object will only be created for Armature Bones which are setup as Deforming Bone types. If a Bone is a Control Bone no Vertex Group will be created on the Child Object for that bone.

To check weather a particular bone in an armature is a Deforming Bone simply switch to Pose Mode or Edit Mode on the armature and select the bone you are interested in by RMB it. Once the bone of interest is selected navigate to [Properties Editor > Bone Context > Deform Panel] and check if the Deform tickable option is ticked or not. If it is the selected bone is a Deforming Bone, otherwise it is a Control Bone. See figure 22.

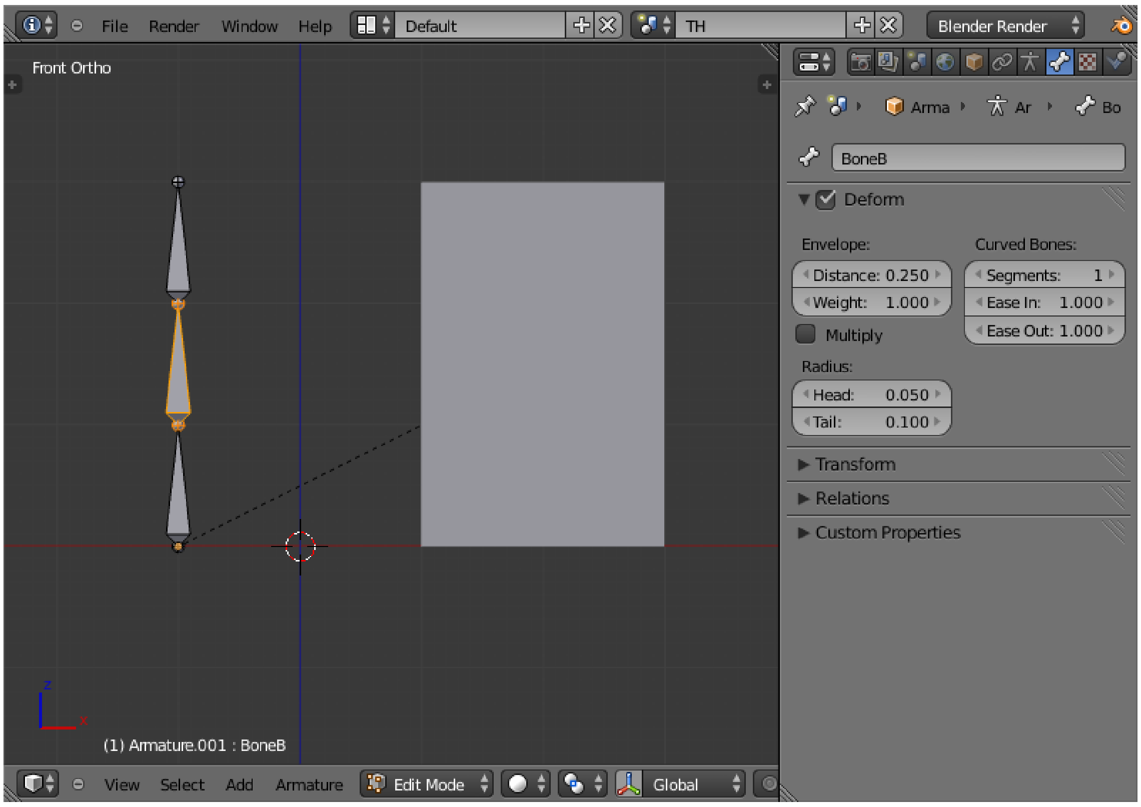


Figure 22 - 3 Bone Armature in Edit Mode with 2nd bone selected with [Properties Editor > Bone Context > Deform Panel] displayed an ticked, indicating the bone is a Deforming Bone.

Armature Deform With Automatic Weights

Armature Deform With Automatic Weights parenting feature does everything Armature Deform With Empty Groups does with one extra thing. That extra thing is that unlike Armature Deform With Empty Groups which leaves the automatically created Vertex Groups empty with no vertices assigned to them; Armature Deform With Automatic Weight will try to calculate how much Influence Weight a particular Armature Bone would have on a certain collection of vertices based on the distance from those vertices to a particular Armature Bone.

Once Blender has calculated the Influence Weight vertices should have it will assign that Influence Weight to the Vertex Groups that were previously created automatically by Blender on the Child Object when Armature Deform With Automatic Weights parenting command was carried out.

If all went well it should be possible to select the Armature Object switch it into Pose Mode and transform the bones of the Armature and the Child Object should deform in response. Unlike Armature Deform parenting you won't have to create Vertex Groups on the Child Object, neither will you have to assign Influences Weights to those Vertex Groups, Blender will try to do it for you.

To activate Armature Deform With Automatic Weights you must be in Object Mode or Pose Mode, then select all the Child Objects (usually Mesh Object Types) and lastly select the Armature Object; Once done press CtrlP and select the Armature Deform With Automatic Weights from the Set Parent To popup dialog.

This method of parenting is certainly easier setup but it can often lead to Armatures which do not deform Child Objects in ways you would want, as Blender can get a little confused when it comes to determining which Bones should influence certain vertices when calculating Influence Weights for more complex armatures and Child Objects. Symptoms of this confusion are that when transforming the Armature Object in Pose Mode parts of the Child Objects don't deform as you expect; If Blender does not give you the results you require you will have to manually alter the Influence Weights of vertices in relation to the Vertex Groups they belong to and have influence in.

You can easily check how much Influence Weight an Armature Bone has over vertices in 2 ways:

- Using the [3D View Editor > Properties Region > Vertex Weights Panel].
- Using Weight Paint Mode.

[3D View Editor > Properties Region > Vertex Weights Panel] - Allows you to see which Vertex Groups a particular vertex is a member of and how much Influence Weight that vertex has in the different Vertex Groups that it is a member of. Remember that Vertex Groups that have the same name as an Armatures Bones will be influenced by the bones in the Armature with the matching names. See figure 23.

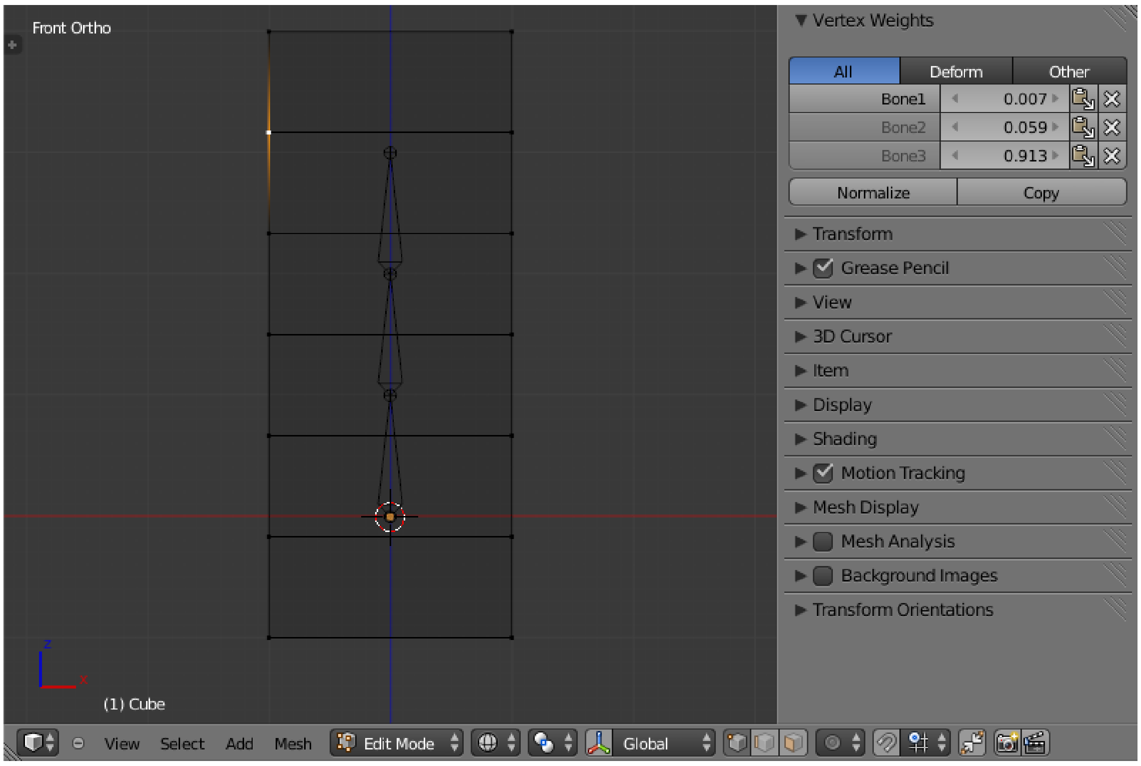


Figure 23 - Armature with 3 Bones and a Mesh Object with a vertex selected which is the Child of the Armature. The [3D View Editor > Properties Region > Vertex Weights Panel] display showing the Influence Weights of the selected vertex.

Weight Paint Mode - Allows you to see the Influence Weights that Vertex Groups have on a particular objects mesh. If does this by color coding a mesh with various colors; These colors represent Influence Weights assigned to certain Vertex Groups.

When a mesh is displayed in Weight Paint Mode, parts of a mesh colored dark blue represent areas of a mesh that are not influenced by a certain Vertex Group, while those parts of the mesh displayed in bright red are influenced by certain Vertex Groups.

Red represents an Influence Weight of 1, blue represents and Influence Weight of 0, and all other colors represent Influence Weights between the 2 values. See figure 24.

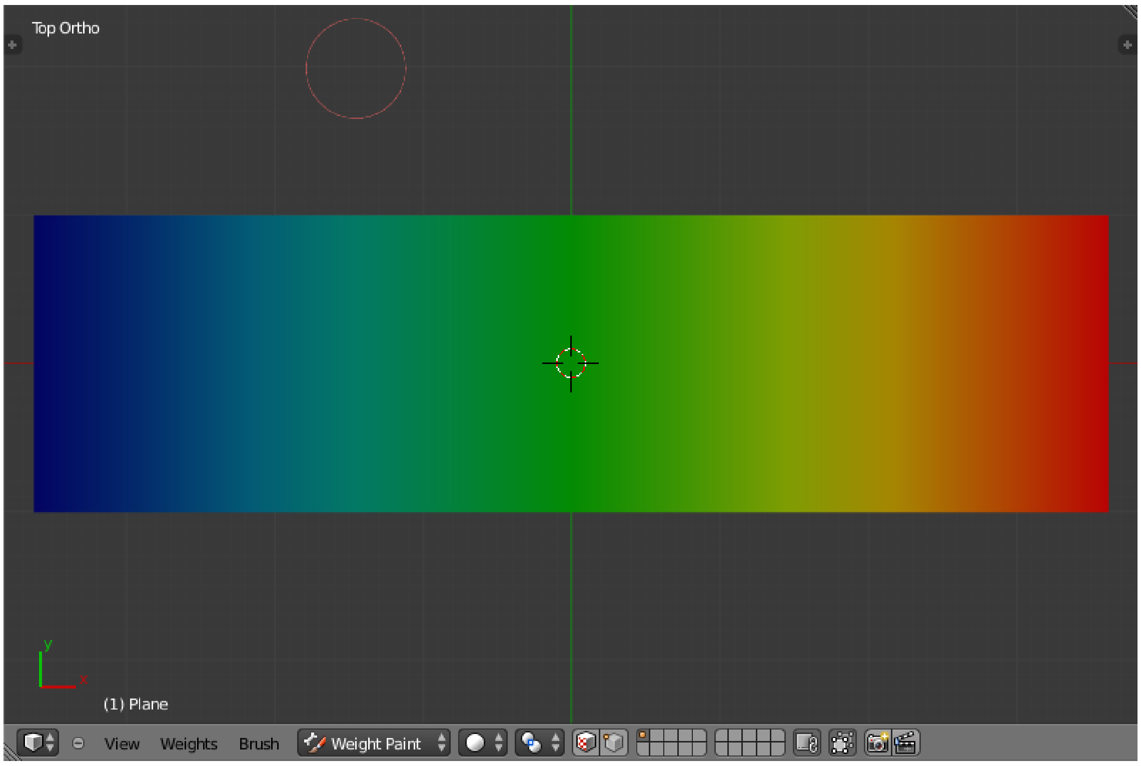


Figure 24 - Plane displayed in Weight Paint Mode showing all the Influence Weight Colors from 0 to 1 with each Influence Weight indicated.

To active Weight Paint Mode select the object you want to see in Weight Paint Mode by LMB it, then navigate to [3D View Editor Header > Mode Select Button] and select the Weight Paint entry. See figure 25.

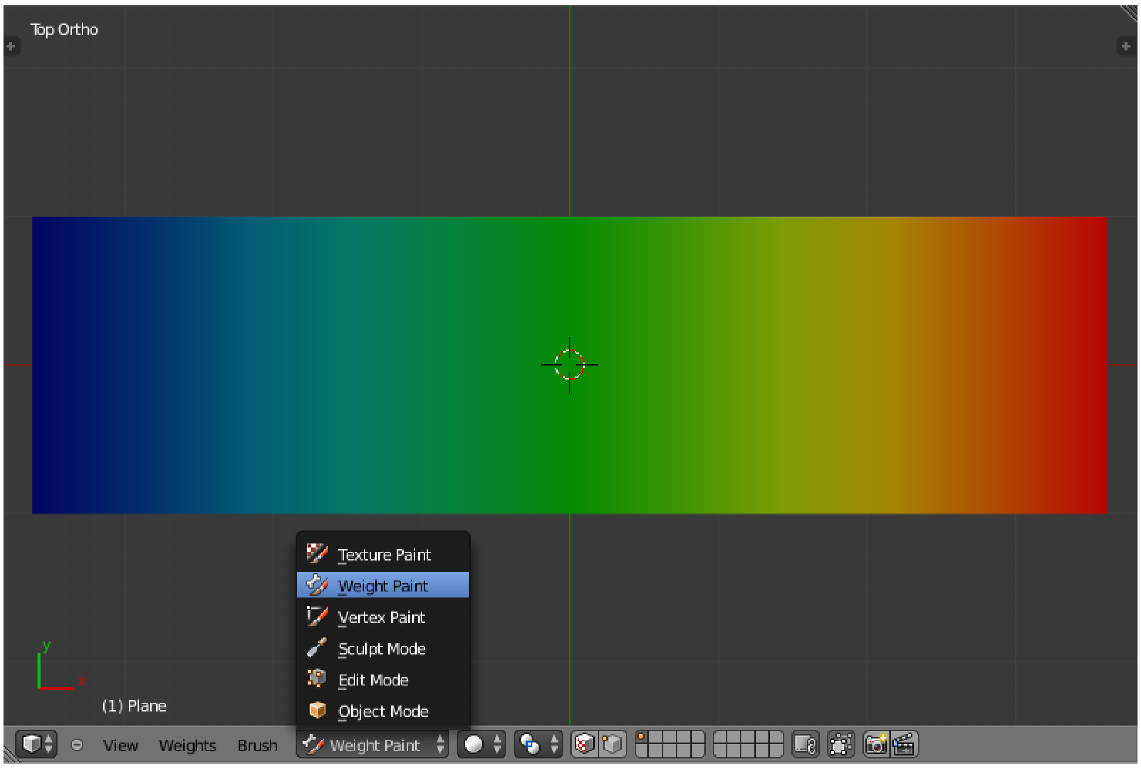


Figure 25 - 3D View Editor Header > Mode Select Button] with Weight Paint Highlighted.

When Weight Paint Mode is used on a Mesh Object which is also the Child Object of an armature, clicking on a bone in the armature will shown the Influence Weight of the selected bone in Pose Mode. See figure 26.

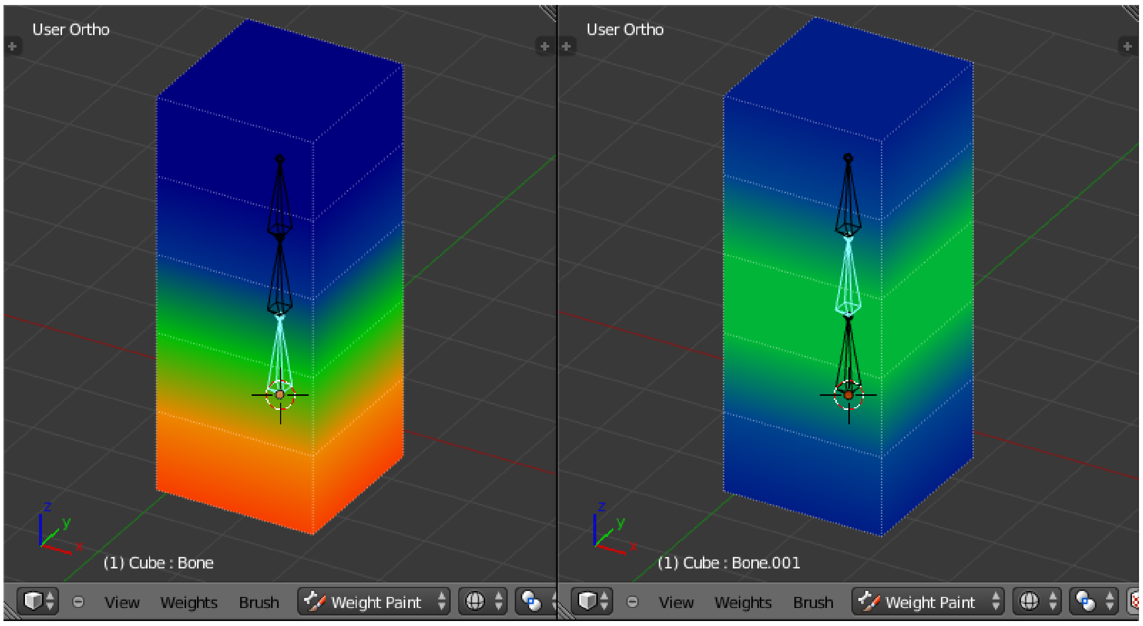


Figure 26 - 3 Bone Armature with a Child Object mesh showing different Weight Paint states when different bones are selected in Pose Mode.

In Weight Paint Mode often you will find that the Armature Bones are obscured from view making them difficult to select; If this happens make sure that only the Armature Object is selected then navigate to [Properties Editor > Object Data Context > Display Panel] and make sure to enable the X-Ray tickable option. This will ensure that the Armature Bones are always visible even when Weight Paint Mode is active and a bone is inside a mesh. See figure 27.

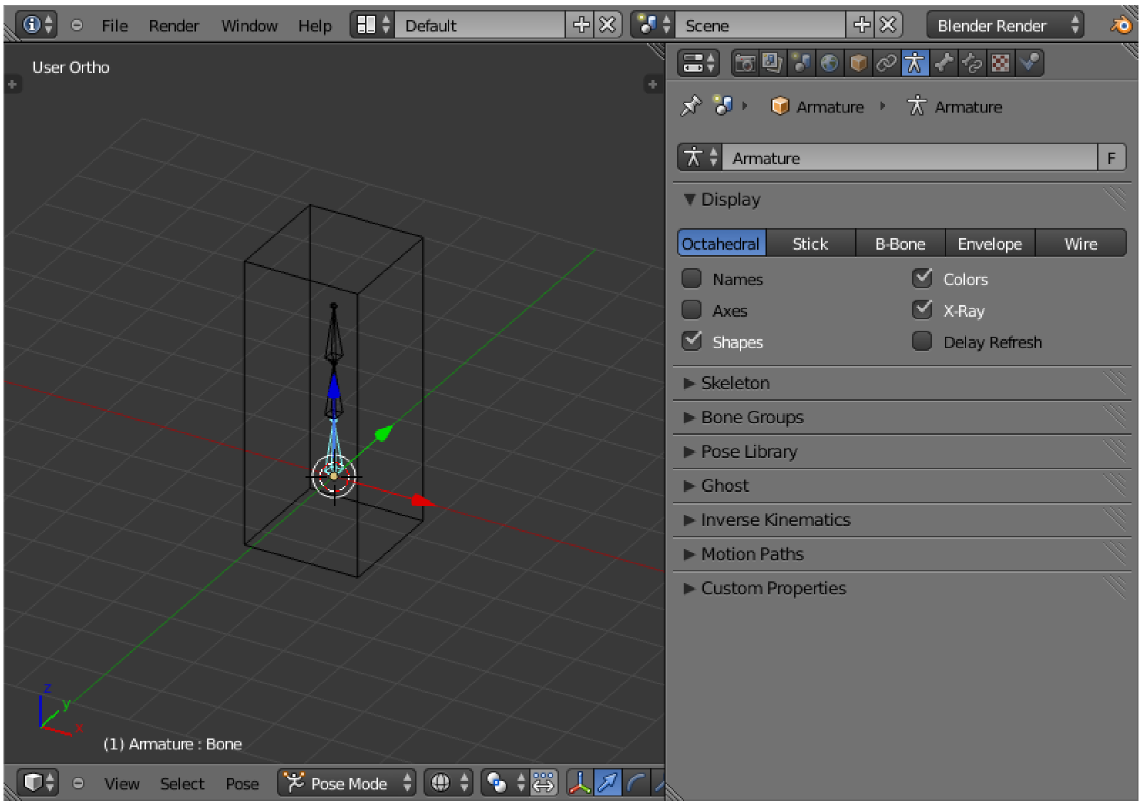


Figure 27 - [Properties Editor > Object Data Context > Display Panel] X-Ray tickable option highlighted.

If you find that a Child Object is deforming strangely when the Armature Bones are manipulated in Pose Mode, using Weight Paint Mode to see visually which parts of a mesh are influenced by a bone is a quick way to try and narrow down which bone is causing the problem.

Armature Deform With Envelope Weights

Works in a similar way to Armature Deform With Automatic Weights in that it will create Vertex Groups on the Child Objects that have names matching those of the Parent Object Armature Bones. The created Vertex Groups will then be assigned Influence Weights. The major difference is in the way those Influence Weights are calculated.

Influence Weights that are calculated when using Armature Deform With Envelope Weights parenting are calculated entirely visually using Bone Envelopes. See figure 28.

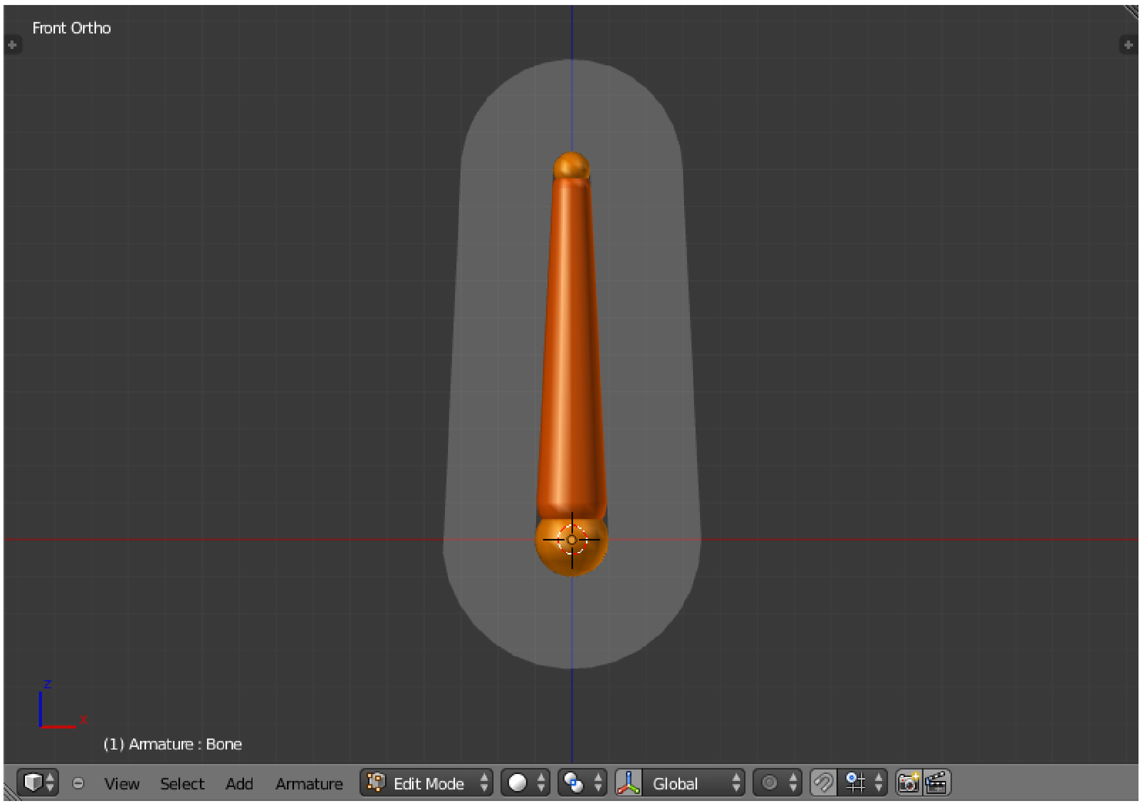


Figure 28 - Single Armature Bone in Edit Mode with Envelope Weight display enabled. The gray volume around the bone is the Bone Envelope.

Figure 28 shows a single Armature Bone in Edit Mode with Envelope Weight activated. The gray semi-transparent volume around the bone is the Bone Envelope.

Any Child Object that has vertices inside the volume of the Bone Envelope will be influenced by the Parent Object Armature when the Armature Deform With Envelope Weights operator is used. Any vertices outside the Bone Evelope volume will not be influenced. See figure 29.

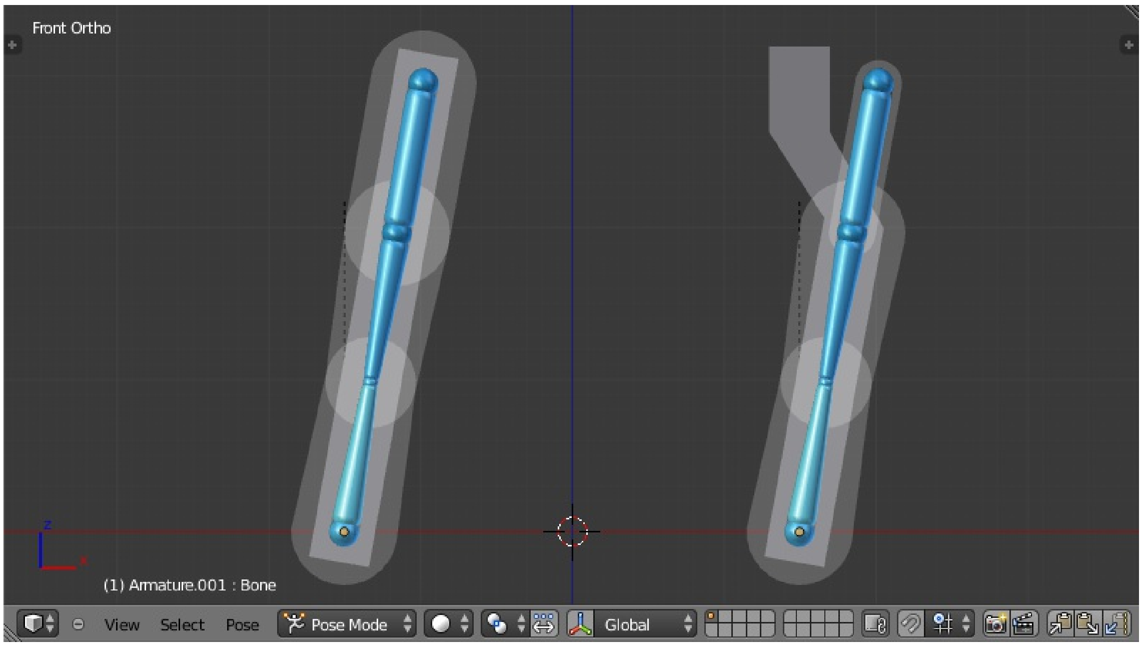


Figure 29 - 2 sets of Armatures each with 3 bones, the first set has all vertices inside the Bone Envelope, the second did not. When the bones are transformed in Pose Mode the results are very different.

The default size of the Bone Envelope volume does not extend very far from the surface of a bone; You can alter the size of the Bone Envelope volume by clicking on the body of the bone you want to alter, switch to Edit Mode or Pose Mode and then pressing CtrlAltS then drag your mouse left or right and the Bone Envelope volume will alter accordingly. See figure 30.

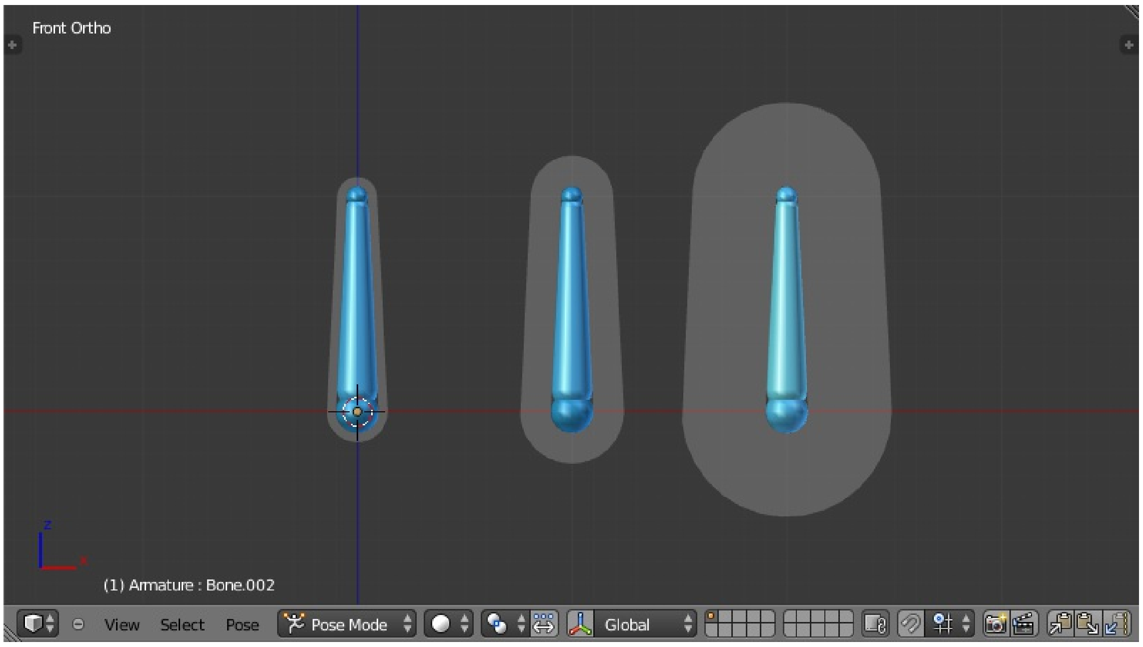


Figure 30 - Single Armature Bone with various different Bone Evelope sizes.

You can also alter the Bone Envelope volume by selecting the Bone you wish to alter and switching to Edit Mode or Pose Mode, then navigate to [Properties Editor > Bone Context > Deform Panel > Envelope Section > Distance field] then enter a new value into it. See figure 31.

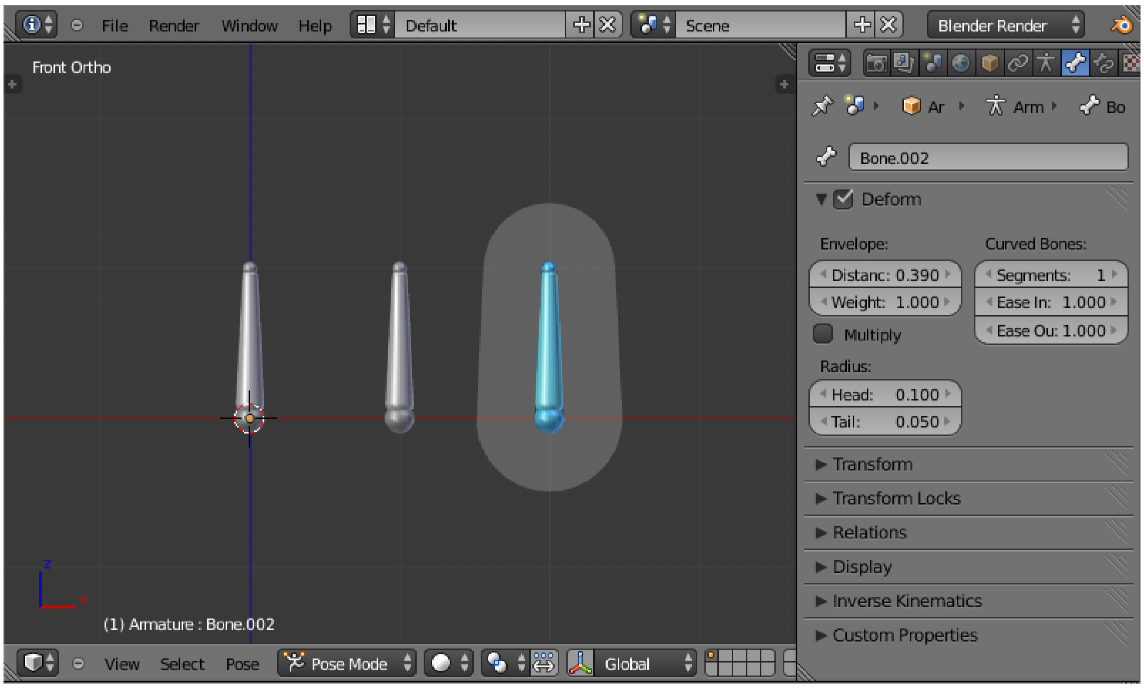


Figure 31 - [Properties Editor > Bone Context > Deform Panel > Envelope Section > Distance field] highlighted.

Altering the Bone Envelope volume does not alter the size of the Armature Bone just the range within which it can influence vertices of Child Objects.

You can alter the radius that a bone has by selecting the head, body or tail parts of a bone while in Edit Mode, and then press AltS and move the mouse left or right. This will make the selected bone fatter or thinner without altering the thickness of the Bone Envelope volume. See figure 32.

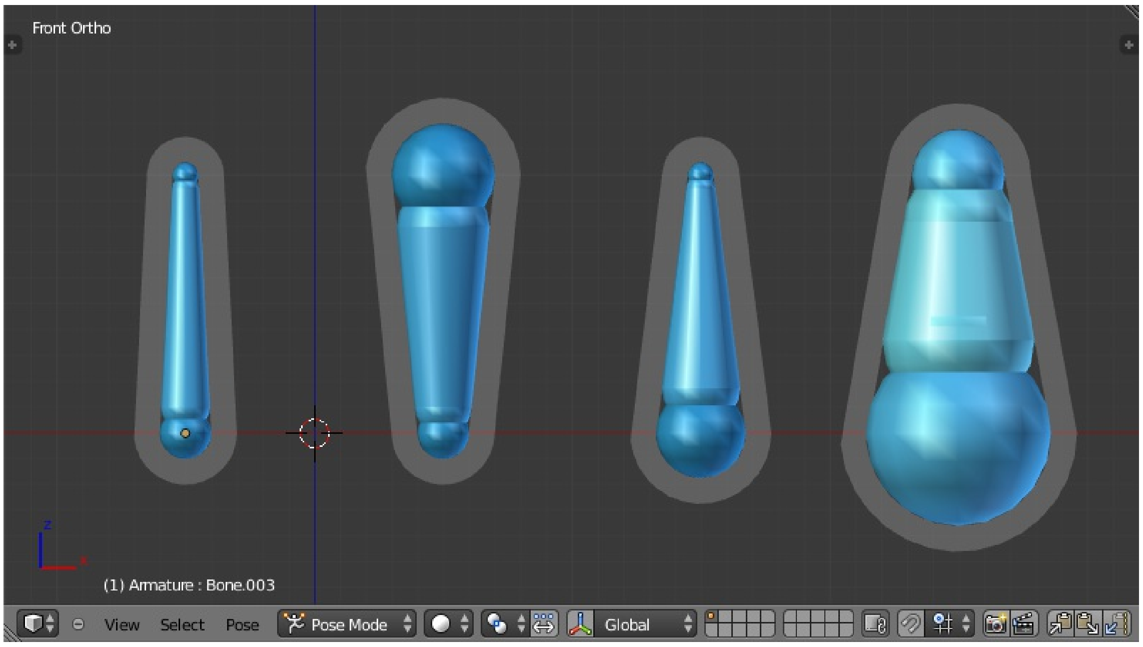


Figure 32 - 4 Armature Bones all using Envelope Weight. The 1st with a default radius value, the 3 others with differing Tail, Head and Body radius values.

You can also alter the bone radius by selecting the tail or head of the bone you wish to alter and switching to Edit Mode, then navigate to [Properties Editor > Bone Context > Deform Panel > Radius Section] and entering new values for the Tail and Head fields. See figure 33.

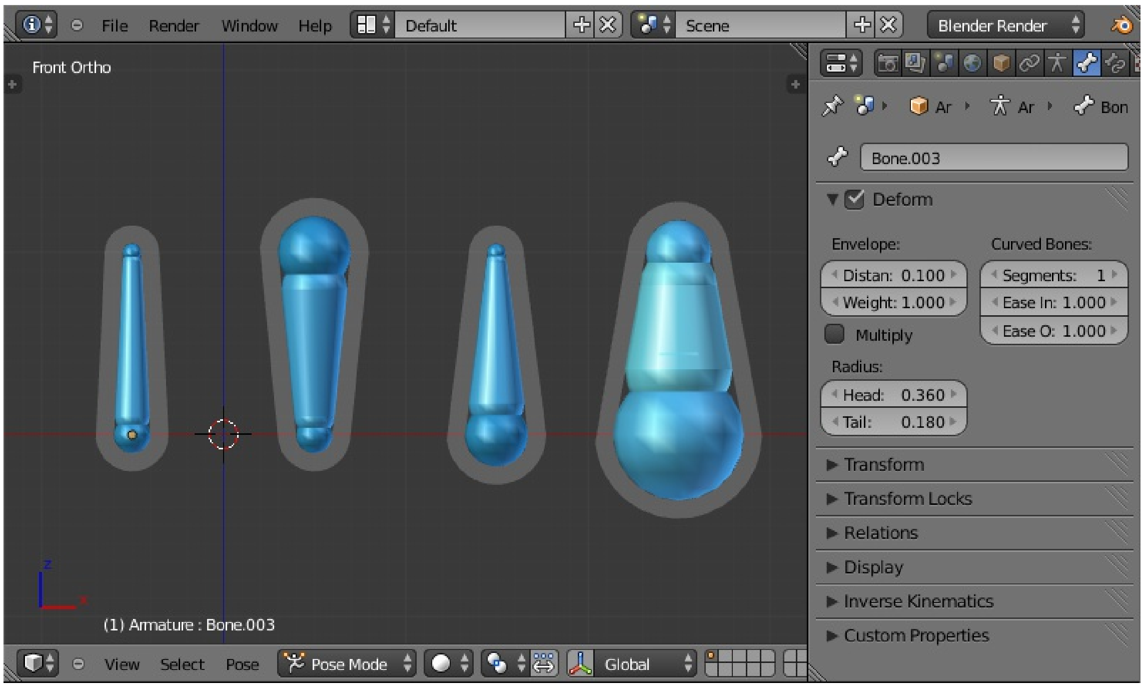


Figure 33 - [Properties Editor > Bone Context > Deform Panel > Radius Section] head and tail fields highlighted.

Note

If you alter the Bone Envelope volume of a bone so that you can have it include/exclude certain vertices after you have already used Armature Deform With Envelope Weights, by default the newly included/excluded vertices wont be effected by the change. When using Armature Deform With Envelope Weights it only calculates which vertices will be affected by the Bone Envelope volume at the time of parenting, at which point it creates the required named Vertex Groups and assigns vertices to them as required. If you want any vertices to take account of the new Bone Envelope volume size you will have carry out the Armature Deform With Envelope Weights parenting again; In fact all parenting used in the Set Parent To popup dialog which tries to automatically assign vertices to Vertex Groups works like this.

Bone Parent

Bone parenting allows you to make a certain bone in an armature the Parent Object of another object. This means that when transforming an armature the Child Object will only move if the specific bone it is the Child Object of moves. See figure 34.

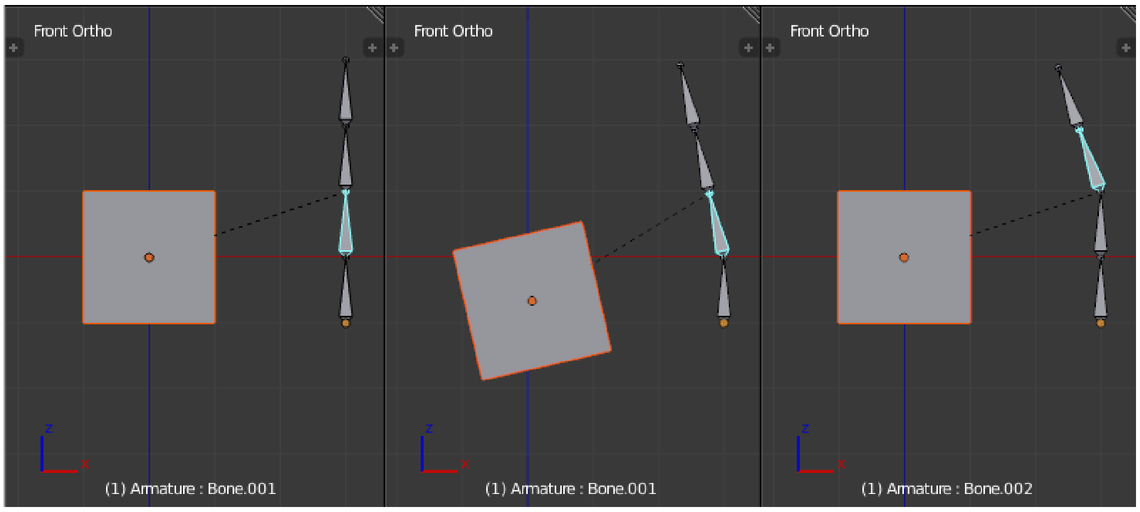


Figure 34 - [3 pictures of Armatures with 4 Bones, with the 2nd bone being the Bone Parent of the Child Object Cube. The Cube is only transformed if the 1st or 2nd bones are. Notice altering the 3rd and 4th bones has no effect on the Cone.

To use Bone Parenting, you must first select all the Child Objects you wish to parent to a specific Armature Bone, then [SHIFT Key+Right Mouse Button Click] select the Armature Object and switch it into Pose Mode and then select the specific bone you wish to be the Parent Bone by [Right Mouse Button Click] it. Once done press [CTRL Key+P Key] and select Bone from the Set Parent To popup dialog.

Now transforming that bone in Pose Mode will result in the Child Objects also transforming.

Bone Parenting - Example Of Use

1. Start Blender with it's default scene.
2. Switch to Front View in the 3D Viewport by pressing [NUMPAD 1 Key].
3. Move the default cube 3 Blender Units to the left on the X axis by pressing [G Key] then [X Key] then input value -3.
4. With the default cube moved, add an Armature Object. Do this by pressing [SHIFT Key+A Key] to display the Add popup menu,

navigate to the Armature entry and select Single Bone.

5. With only the single Armature Bone selected switch into Edit Mode by pressing the [TAB Key].
6. Select the top pointy end of the bone (the Tail).
7. Extrude a new bone from the tail of the old bone that is 1 Blender Unit in length along the Z axis. Do this by pressing the [E Key] then the [Z Key] then input the value 1, then press the [ENTER Key].
8. Create 2 more bones each extruding out of the Tail of the previous bone. Do this by pressing [SHIFT Key+R Key] twice; This will execute the Repeat Last command operator.
9. Switch back into Object Mode by pressing the [TAB Key].
10. You should now have a default cube and an Armature Object with 4 bones positioned as shown in figure 35].

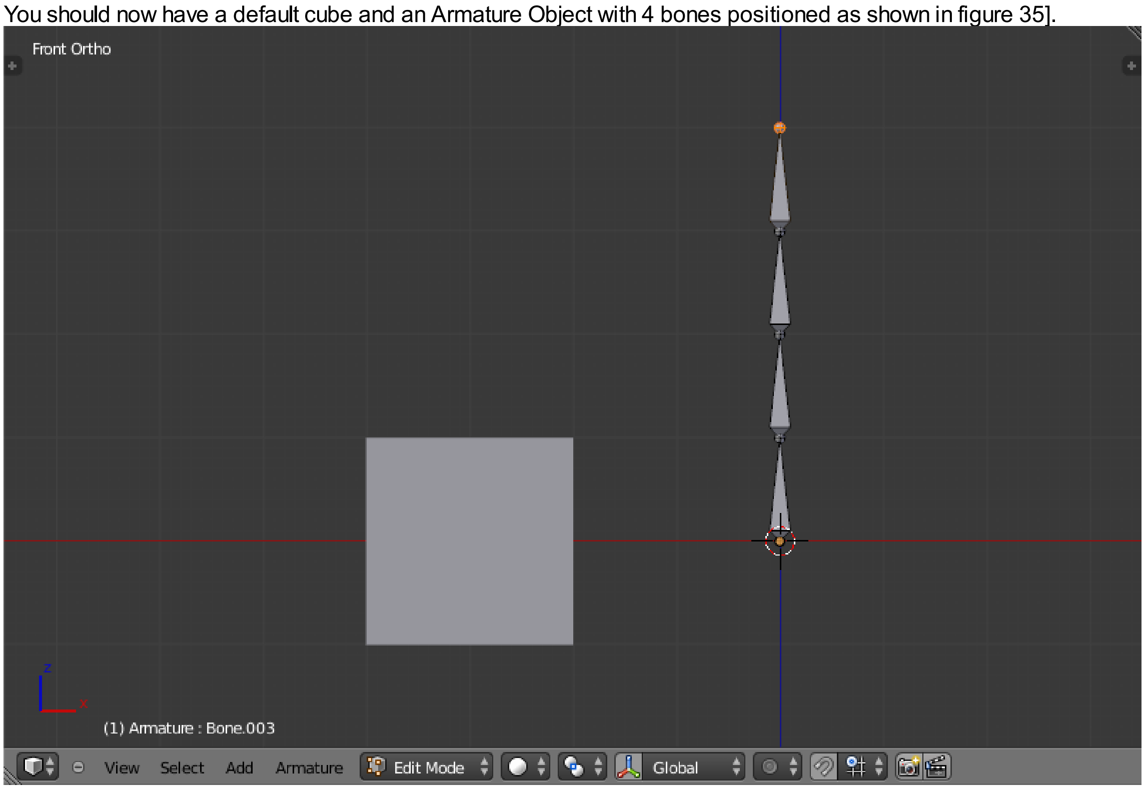


Figure 35 - [Default cube and Armature Object with 4 bones.

11. Select the default cube only and then [SHIFT Key+Right Mouse Button Click] the Armature Object.
12. With the Cube and the Armature Object still selected navigate to [3D View Editor Header > Mode Select Button] and switch to Pose Mode.
13. [Right Mouse Button Click] on the second bone in the Armature Object, such that it becomes highlighted in Cyan. See figure 36.

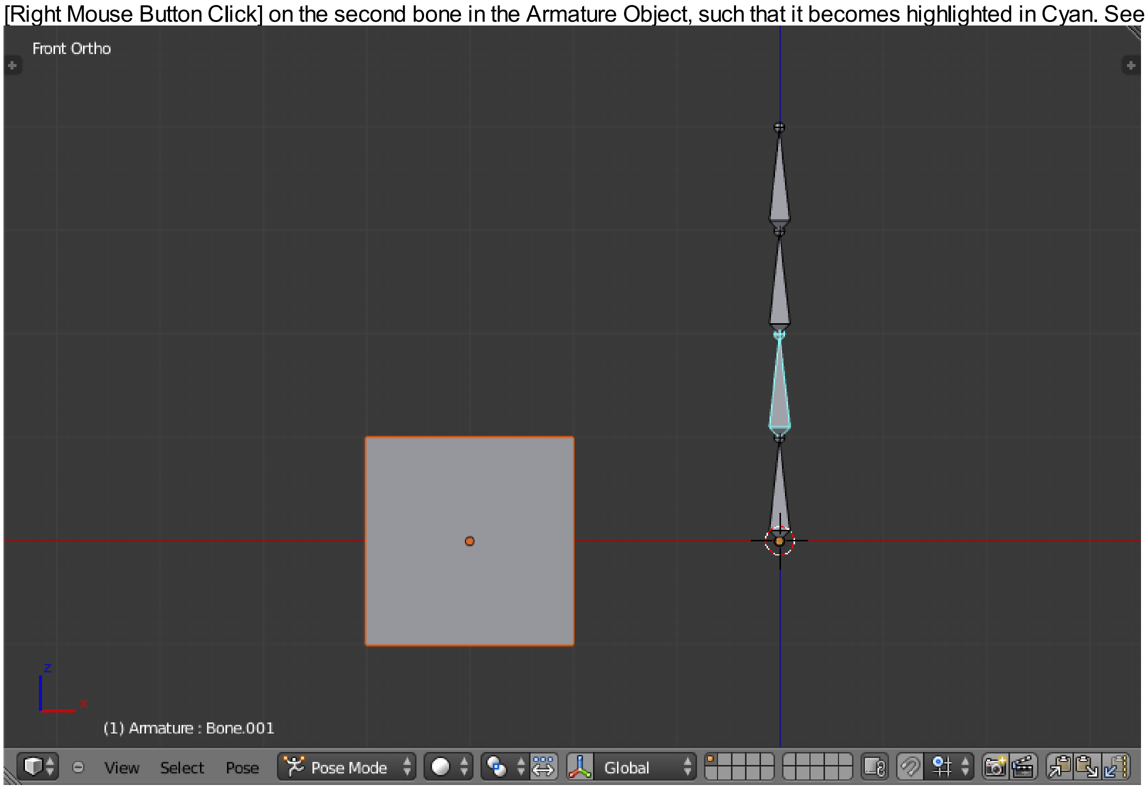


Figure 36 - [Default Cube selected and the Armature Object showing in Pose Mode with second bone selected and displayed in Cyan.

14. Now we need to make that selected Armature Bone the Parent Bone of the Cube Child Object. To do that activate the Set Parent To popup dialog by pressing [CTRL Key+P Key] and selecting the Bone entry from the popup.
15. Once done switch back into Object Mode and select only the Armature Object.
16. Now if you switch the Armature Object back into Pose Mode and transform the 3rd or 4th bones of the armature the Cube Child Object is not affected, but if you transform the 1st or 2nd bones it is because altering either of those bones results in the 2nd

bone being transformed.

Bone Relative Parenting

Bone Relative parenting works in the same way as Bone parenting with one difference.

With Bone parenting if you have parented a bone to some Child Objects and you select that bone and switch it into Edit Mode and then translate that bone; When you switch back into Pose Mode on that bone, the Child Object which is parented to that bone will snap back to the location of the bone in Pose Mode. See figure 37.

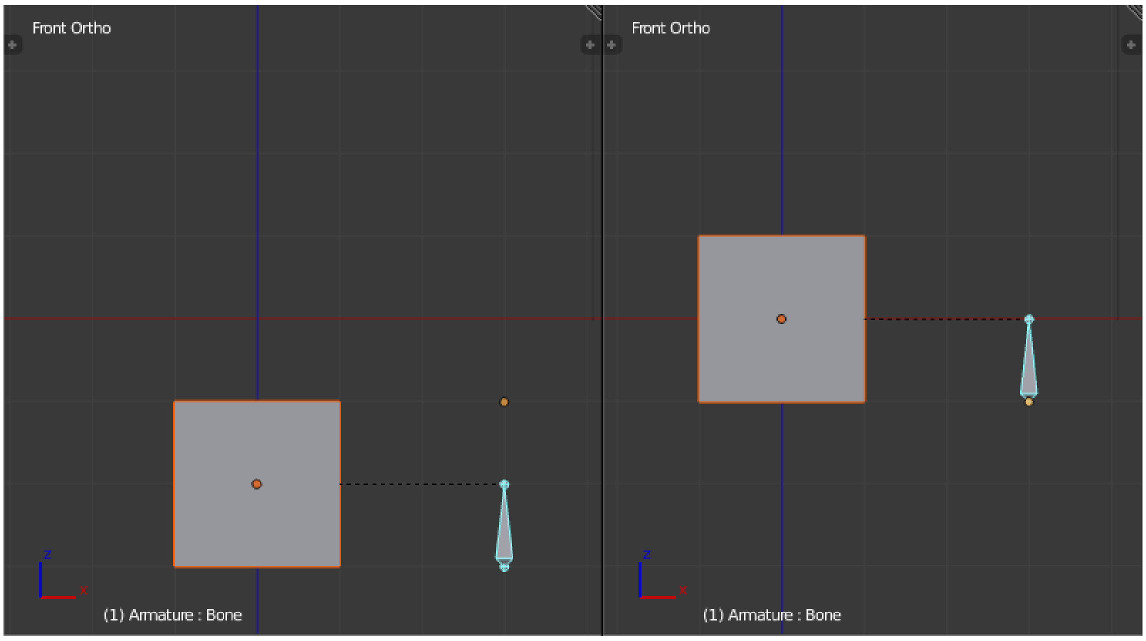


Figure 37 - [Single Armature Bone which has a Child Object cube parented to it using Bone parenting. 1st picture shows the position of the cube and armature before the bone is moved in Edit Mode. 2nd picture shows the position of the cube and armature after the bone was selected in Edit Mode, moved and switched back into Pose Mode. Notice that the Child Object moves to the new location of the Pose Bone.

Bone Relative parenting works differently; If you move a Parent Bone in Edit Mode, when you switch back to Pose Mode, the Child Objects will not move to the new location of the Pose Bone. See figure 38.

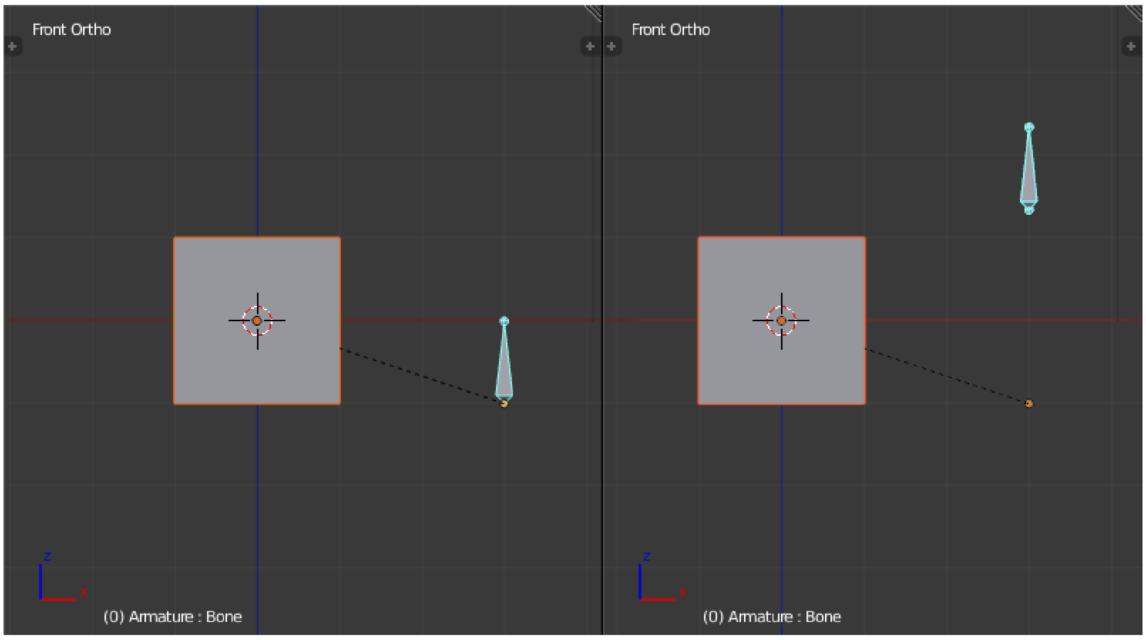


Figure 38 - [Single Armature Bone which has a Child Object cube parented to it using Bone Relative parenting. 1st picture shows the position of the cube and armature before the bone is moved in Edit Mode. 2nd picture shows the position of the cube and armature after the bone was selected in Edit Mode, moved and switched back into Pose Mode. Notice that the Child Object does not move to the new location of the Pose Bone.

Vertex Parent

You can parent an object to a single vertex or a group of three vertices as well; that way the child/children will move when the parent mesh is deformed, like a mosquito on a pulsing artery.

Vertex Parent from Edit Mode

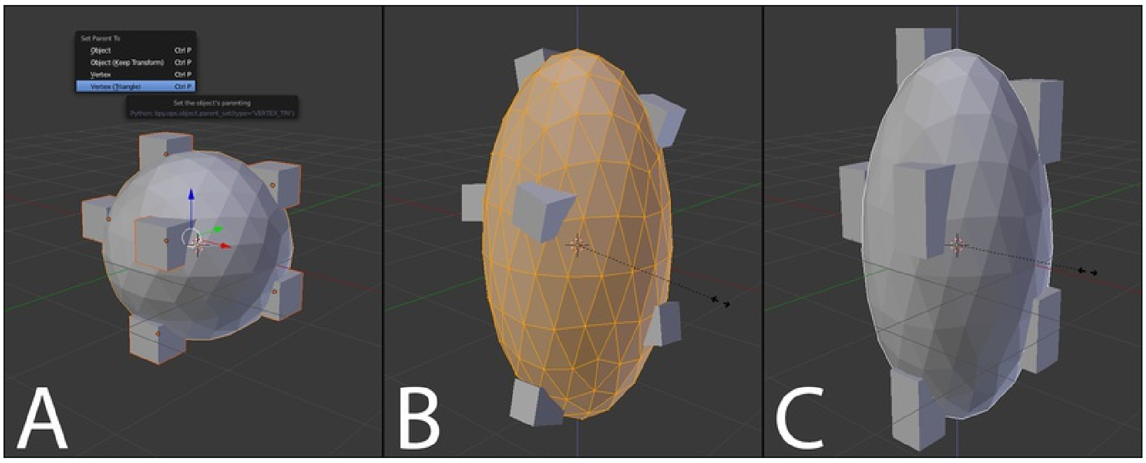
In Object mode, select the child/children and then the parent object. ⇆ Tab into Edit mode and on the parent object select either one vertex that defines a single point, or select three vertices that define an area (the three vertices do not have to form a complete face; they can be any three vertices of the parent object), and then hit CtrlP and confirm.

At this point, if a single vertex was selected, a relationship/parenting line will be drawn from the vertex to the child/children. If three vertices were selected then a relationship/parenting line is drawn from the averaged center of the three points (of the parent object) to the child/children. Now, as the parent mesh deforms and the chosen parent vertex/vertices move, the child/children will move as well.

Vertex Parent from Object Mode

Vertex parenting can be performed from object mode, This is done like regular object parenting, Press CtrlP in object mode and select Vertex or Vertex (Triangle).

The nearest vertices will be used from each object which is typically what you would want.



A) The small cubes can each be automatically parented to a triad of nearby vertices on the icosphere using the "Vertex (Triangle)" in the set parent context menu.
B) Reshaping the object in edit mode then means each of the cubes follows their vertex triad parent separately.
C) Re-scaling the parent icosphere in object mode means the child cubes are also rescaled as expected.

The parent context menu item means users can rapidly set up a large number of vertex parent relationships, and avoid the tedious effort of establishing each parent-child vertex relationship separately.

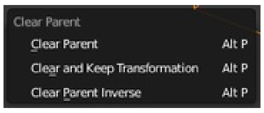
Note
It is in fact a sort of “reversed” hook

Options

Move child

You can *move* a child object to its parent by clearing its origin. The relationship between the parent and child isn’t affected. Select the child object and press AltO. By confirming the dialog the child object will snap to the parent’s location. Use the Outliner view to verify that the child object is still parented.

Remove relationship/Clear Parent


Clear Parent
Clear Parent Alt P
Clear and Keep Transformation Alt P
Clear Parent Inverse Alt P

Clear Parent pop-up menu

You can *remove* a parent-child relationship via AltP

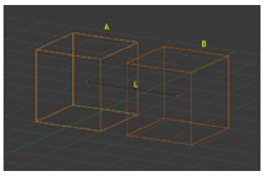
The menu contains:

Clear Parent
: If the parent in the group is selected nothing is done. If a child or children are selected they are disassociated from the parent, or freed, and they return to their *original* location, rotation, and size.

Clear and Keep Transformation
: Frees the children from the parent, and *keeps* the location, rotation, and size given to them by the parent.

Clear Parent Inverse
: Places the children with respect to the parent as if they were placed in the Global reference. This effectively clears the parent’s transformation from the children. For example, if the parent is moved 10 units along the X axis and Clear Parent Inverse is invoked, any selected children are freed and moved -10 units back along the X axis. The “Inverse” only uses the last transformation; if the parent moved twice, 10 units each time for a total of 20 units, then the “Inverse” will only move the child back 10 units, not 20.

Parenting Example

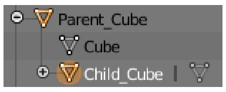


Parenting Example

The active object, in yellow (cube `A`), will be made the parent of all the other object(s) in the group (orange cube `B`). The center(s) of all children object(s) are now linked to the center of the parent by a dashed line; see image (*Parenting Example*). The parent object is cube “`A`” and the child object is cube “`B`”. The link is labeled “`L`”.

At this point, grabbing, rotating, and scaling transformations to the parent will do the same to the children. Parenting is a very important tool with many advanced applications, as we’ll see in later chapters; it is used extensively with advanced animations.

Hints



Outliner view

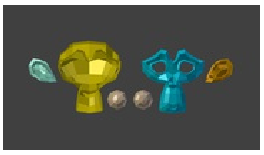
There is another way to see the parent-child relationship in groups and that is to use the Outliner view of the Outliner window. Image (*Outliner view*) is an example of what the Outliner view looks like for the (*Parenting Example*). Cube “`A`”’s object name is “`Cube_Parent`” and cube “`B`” is “`Cube_Child`”.

Separating Objects

At some point, you’ll come to a time when you need to cut parts away from a mesh to be separate. Well, the operation is easy.

To separate an object, the vertices (or faces) must be selected and then separated, though there are several different ways to do this. In Edit Mode, press P then select one of the following.

Options

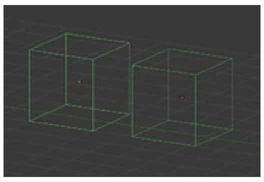

Suzanne dissected neatly

Selected
: This option separates the selection to a new object.

All Loose Parts
: Separates the mesh in its unconnected parts.

By Material
: Creates separate mesh objects for each material.

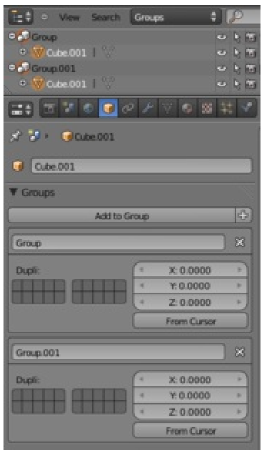
Grouping objects


Grouped objects

Group objects together without any kind of transformation relationship. Use groups to just logically organize your scene, or to facilitate one-step appending or linking between files or across scenes. Objects that are part of a group always shows as light green when selected; see image (*Grouped objects*).

Options

Creating a Group
: CtrlG creates a new group and adds the selected object(s) to it.



Naming a Group

Naming a Group
: All groups that an object has been assigned to are listed in the Object Properties Panel's Relations panel. To rename a group, simply click in the groups name field.
: To name groups in the Outliner window, select Groups as the outliner display from the header combo box, and Ctrl LMB click on the group name. The name will change to an editable field; make your changes and press ↵ Enter.

Restricting Group Contents via Layers
: The cluster of layer buttons attached to each group determines from which layers the group objects will be included when duplicated. If your group contains objects on layers 10, 11 and 12, but you disable the layer 12 button in the group controls, duplicates of that group (using the Dupligroup feature) will only show the portions of the group that reside in layers 10 and 11.

Appending or Linking Groups
: To append a group from another .blend file, consult this page. In summary, File → Append or Link → `filename` → Group → `groupname`.

Removing Groups
: To remove a object from a group, under the object context button, open the "Groups" pane. Find the name of the group from which you wish to remove the object, and click the x to the right of the group name.

Select Grouped

Mode: Object mode

Hotkey: ⇧ ShiftG

Menu: Select → Grouped

Select objects by parenting and grouping characteristics. See Select Grouped for more information.

Duplication

Mode: Edit and Object modes

Hotkey: ⇧ ShiftD

Menu: Object → Duplicate

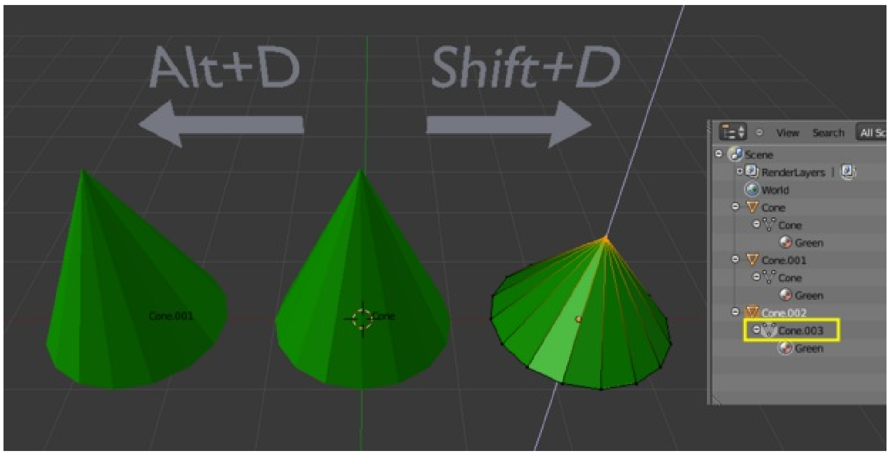
Description

This will create a visually-identical copy of the selected object(s). The copy is created at the same position as the original object and you are automatically placed in Grab mode. Reference (*Duplicate Example*) for the discussions below.

This copy is a new object, which "**shares**" some datablocks with the original object (by default, all the Materials, Textures, and Ipos), but which has copied others, like the mesh, for example. This is why this form of duplication is sometimes called "shallow link", because not all datablocks are shared; some of them are "hard copied"!

Note that you can choose which types of datablock will be linked or copied when duplicating: in the User Preferences' (available in the File menu) Editing "tab", activate those types of datablocks you want to really copy in the Duplicate Data list — the others will just be linked.

Examples



The mesh Cone.006 of object Cone.002 is being edited. The mesh's Unique datablock ID name is highlighted in the Outliner.

The cone in the middle has been (1) link duplicated to the left and (2) duplicated to the right.

- The duplicated right cone is being edited, the original cone in the middle remains unchanged. The mesh data has been copied, not linked.
- Likewise, if the right cone is edited in object mode, the original cone remains unchanged. The new object's transform properties or datablock is a copy, not linked.
- When the right cone was duplicated, it inherited the material of the middle cone. The material properties were linked, not copied.

See above if you want separate copies of the datablocks normally linked.

Linked Duplicates

Mode: Object mode

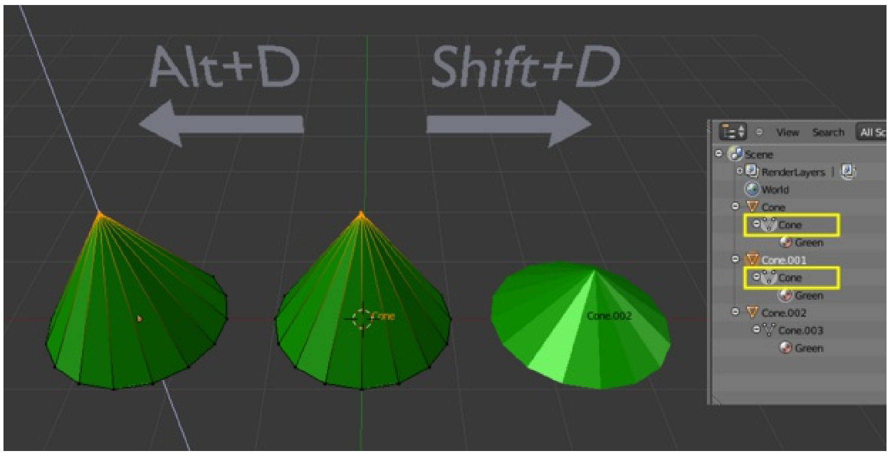
Hotkey: AltD

Menu: Object → Duplicate Linked

Description

You also have the choice of creating a *Linked Duplicate* rather than a *Duplicate*; this is called a deep link. This will create a new object with **all** of its data linked to the original object. If you modify one of the linked objects in Edit mode, all linked copies are modified. Transform properties (object datablocks) still remain copies, not links, so you still can rotate, scale, and move freely without affecting the other copy. Reference (*Duplicate Example*) for the discussions below.

Examples



The object Cone.001 was linked duplicated. Though both these cones are separate objects with unique names, the single mesh named Cone, highlighted in the Outliner, is shared by both.

The left cone is a Linked Duplicate of the middle cone (using AltD).

- As a vertex of one object is moved in Edit mode, the same vertex is moved in the original cone as well. The mesh data are links, not copies.
- In contrast, if one of these two cones is rotated or rescaled in object mode, the other remains unchanged. The transform properties are copied, not linked.
- As in the previous example, the newly created cone has inherited the material of the original cone. The material properties are linked, not copied.

A common table has a top and four legs. Model one leg, and then make linked duplicates three times for each of the remaining legs. If you later make a change to the mesh, all the legs will still match. Linked duplicates also apply to a set of drinking glasses, wheels on a car… anywhere there is repetition or symmetry.

Procedural Duplication

Mode: Object mode and Edit mode

Panel: Object settings

There are currently four ways in Blender to procedurally duplicate objects. These options are located in the Object menu.

Verts
: This creates an instance of all children of this object on each vertex (for mesh objects only).

Faces
: This creates an instance of all children of this object on each face (for mesh objects only).

Group
: This creates an instance of the group with the transformation of the object. Group duplicators can be animated using actions, or can get a Proxy.

Frames
: For animated objects, this creates an instance on every frame. As you'll see on this topic's subpage, this is also a *very* powerful technique for arranging objects and for modeling them.

Linked Library Duplication

Hotkey: ⇧ ShiftF1

Menu: File → Link Append

Linked Libraries
: Linked Libraries are also a form of duplication. Any object or datablock in other .blend files can be reused in the current file.

Hints

- If you want transform properties (i.e. object datablocks) to be "linked", see the page on parenting.
- Material Transparency will not display when instancing dupli-groups; this is a known limitation of Blender's view-port.

DupliVerts

Mode: Object mode

Panel: Object → Duplication

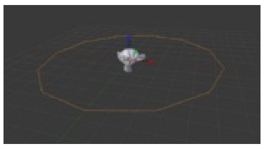
Duplication Vertices or DupliVerts is the duplication of a base object at the location of the vertices of a mesh. In other words, when using DupliVerts on a mesh, an instance of the base object is placed on every vertex of the mesh.

There are actually two approaches to modeling using DupliVerts. They can be used as an arranging tool, allowing us to model geometrical arrangements of objects (e.g. the columns of a Greek temple, the trees in a garden, an army of robot soldiers, the desks in a classroom). The object can be of any object type which Blender supports. The second approach is to use them to model an object starting from a single part of it (e.g. the spikes in a club, the thorns of a sea-urchin, the tiles in a wall, the petals in a flower).

Download example .blend file
You can download a file with the examples described on this page. In this .blend, the first example, a monkey parented to a circle is on layer 1; while a tentacle parented to an icosphere is on layer 2. The files was made using Blender 2.55.1 (r33567).

DupliVerts as an Arranging Tool

Setup


A monkey head and a circle

All you need is a base object (e.g. the *tree* or the *column*) and a pattern mesh with its vertices following the pattern you have in mind. In this section, we will use a simple scene for the following part. We'll be using a monkey head located at the origin of the coordinate system as our base object and a circle at the same location as our parent mesh.

Dupliverted monkeys

First, in Object mode, select the base object and ⇧ Shift RMB to add the circle to the selection (order is very important here), and CtrlP to parent the base object to the circle. Now, the circle is the parent of the monkey; if you move the circle, the monkey will follow it. It is crucial that the base object lies within the confines of the parent mesh (as pictured in *A monkey head and a circle*) in order to achieve the desired appearance of vertex duplication on the parent mesh.

With only the circle selected, enable Duplication vertices in the Object panel→ Duplication→ Verts. A monkey head should be placed at every vertex of the circle.

The original monkey head at the center and the parent mesh are still shown in the 3D view but neither will be rendered. If the placement and rotation of your monkey head is odd, you might need to clear its rotation (AltR), scale AltS, location AltG, and origin AltO.

Rearranging

If you now select the base object and modify it in either object or edit mode, all changes will also affect the shape of all duplicate objects. You can also select the parent mesh to modify the arrangement of the duplicates; adding vertices will also add more base objects. Note that the base objects will inherit changes made to the parent mesh in object mode, but not in edit mode — so scaling the circle up in object mode will enlarge the monkey head, while scaling the circle up in edit mode will only increase the distance between the base objects.

Orientation

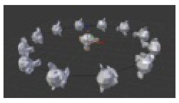
Orientation enabled, orientation +Y

The orientation of the base objects can be controlled by enabling Rotation in the Duplication panel. This will rotate all base objects according to the vertex normals of the parent mesh.

To change the orientation of the duplicated objects, select the base object and in the Object→ Relations extras panel change the

Tracking Axes.

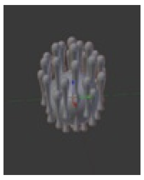
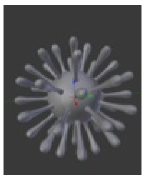
Output of various orientations

		
Negative Y	Positive X	Positive Z, up X

Note
The axes of an object can be made visible in the Object→ Display panel.
To display the vertex normals of the parent mesh, tab into edit mode and enable this function in Properties (N)→ Mesh Display panel where you can also resize the displayed normals as necessary.

DupliVerts as a Modeling Tool

Very interesting models can be made using DupliVerts and a standard primitive. In this example, a simple tentacle was made by extruding a cube a couple of times. The tentacle object was then parented to an icosphere. With dupli Rotation enabled for the parent mesh (the icosphere), the orientation of the base object (the tentacle) was adapted to the vertex normals of the parent mesh (in this case the tentacle was rotated -90° about the X axis in edit mode).

		
A simple tentacle set to smooth	Tentacle dupliverted onto the parent mesh	Rotation enabled to align duplicates

As in the previous example, the shape and proportions of the arrangement can now be tweaked.

To turn all duplicates into real objects, simply select the icosphere and Object→ Apply→ Make Duplicates Real (Ctrl⇧ ShiftA). To make the icosphere and the tentacle a single object, make sure they are all selected and go to Object→ Join (CtrlJ).

See also

Other duplication methods are listed here.

DupliFaces

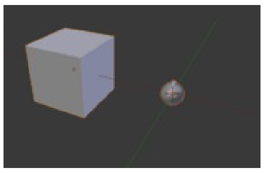
Mode: Object mode

Panel: Object → Duplication

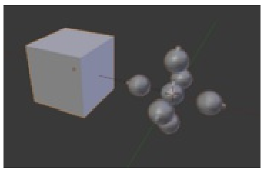
Duplication Faces or DupliFaces is the capability to replicate an object on each face of a parent object. One of the best ways to explain this is through an example illustration.

Example .blend file
Download the .blend file used for the examples on this page here

Basic usage


A cube and a sphere

In this example we will use a UV sphere with an extruded "north pole" as our base object and cube as our parent mesh. To parent the sphere to the cube, in Object mode, first RMB select the sphere, then ⇧ Shift RMB select the cube (order is very important here), and finally CtrlP to parent.

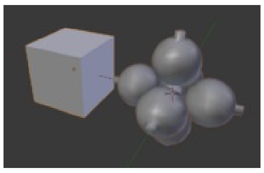

Duplication Faces applied to the cube

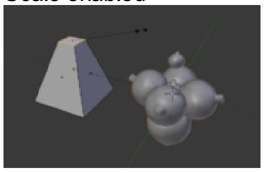
Next, in the Object context's Duplication panel, enable Faces. The sphere is duplicated one for each face of the cube.

Inherited properties
The location, orientation, and scale of the duplicated child(ren) matches that of the faces of the parent. So, if several objects are parented to the cube, they will all be duplicated once for each face on the cube. If the cube is subdivided (in Edit Mode W), every child will be duplicated for each face on the cube.

Both the parent object and original are displayed as editable "templates" in 3D view, but neither is rendered.

Scale


Scale enabled


Top face of cube scaled down

By enabling Scale for the parent object, the scale of the child objects will be adapted to the size of each face in the parent object.

Thus, by rescaling the face of the parent object, the size of the duplicated object will change accordingly.

Limitations / Considerations

The positioning of the duplicated geometry relative to the face is dependent upon the position of the child objects relative to the duplicator's origin. This can lead to some visual artifacts in the editor as the geometry of the original objects overlaps or intersects with the duplicates. One workaround is to move the origin of the duplicator mesh off of the plane of the faces.

If the geometry of the children is not symmetrical then the orientation of the face (as determined by the order of its vertices) could matter. As of 2.70 blender does not have tools which allow you to adjust the ordering of the vertices on a face.

However, there is a workflow that lets you control for this. Make a single square and enable the Duplication / Faces so you can see the duplicated geometry in your editor window. If the orientation is not what you want, rotate the face until it is how you want. Typically you want to do the rotation in Edit mode, not Object mode, but this is not a hard rule.

Once you have the orientation correct, Duplicate the face and move the duplicate where you want it. Repeat this process until you have enough faces. Since it is common for these faces to butt up against one another, your geometry will have lots of duplicate vertices. Use the Remove Doubles button in the Tools panel.

A short video illustrating this workflow.

DupliGroup

Mode: Object mode

Panel: Object → Duplication → Group

Duplication Group or DupliGroup allows you to create an instance of a group for each instance of another object.

Basic Usage

- Create a number of objects and group them by
  1. selecting them all,
  2. CtrlG, and
  3. eventually rename your group in Object → Groups
- Create a DupliGroup by
  1. adding another object (⇧ ShiftA), say an Empty,
  2. in Object → Duplication enable Group, and
  3. select the name of your newly created group in the selection box that appears.

DupliGroup and Dynamic Linking

See Appending and Linking to understand how to dynamically link data from another .blend file into the current file. You can dynamically link groups from one blend file to another. When you do so, the linked group does not appear anywhere in your scene until you create an object controlling where the group instance appears.

Example

- Link a group from another file into your scene, as described in Appending and Linking.
  From here, you can use the easy way or the hard way:
- The easy way:
  1. Select Add → Group Instance → `[name of group you just linked]`.
- The hard way:
  1. Select Add → Empty, and select the empty that you added.
  2. Switch to the Object context, and in the Duplication panel, click Group.
  3. In the dropdown box that appears next to Group:, pick the group that you linked.

At this point, an instance of the group will appear. You can duplicate the empty, and the DupliGroup settings will be preserved for each empty. This way, you can get multiple copies of linked data very easily.

Making a DupliGroup Object Real

Say you want to make further edits on an DupliGroup instance or render the DupliGroup in Yafaray or some other render that does not support importing DupliGroups directly:

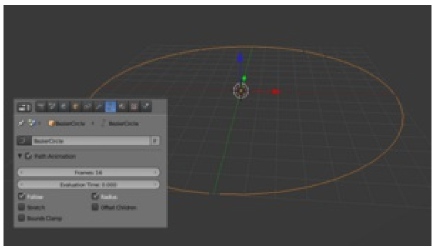
Simply select your DupliGroup and press Ctrl⇧ ShiftA to convert the DupliGroup into regular objects that can be transformed and animated normally.

Note
Note that if the DupliGroup was linked from an external file the Object Data (mesh, materials, textures, transforms) will also still be linked from the original group. However, the various object's parent-child relationships do not carry over.

DupliFrames

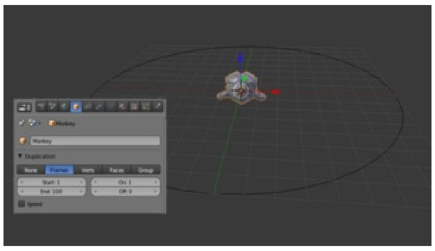
DupliFrames is a tool to duplicate objects at frames distributed along a path. This is a useful tool to quickly arrange objects.

Examples



Settings for the curve

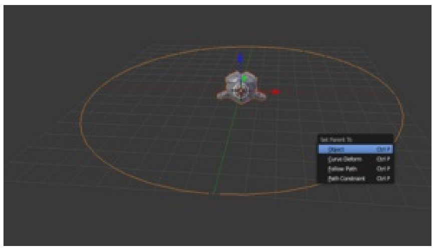
⇧ ShiftA to add a Bezier Circle and scale it up. In the Curve menu under Path Animation enable Follow and set Frames to something more reasonable than 100 (say 16).



Settings for the object

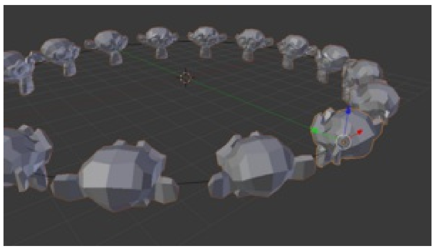
Add a Monkey. In the Object menu under Duplication enable Frames and disable Speed.

Speed
The Speed option is used when the parent-child relationship is set to Follow Path (see below). In this example, the monkey will then travel along the circle over 16 frames.



Parenting

To parent the monkey to the Bezier circle, first select the monkey then the curve (so that the curve is the active object) and CtrlP. Select the monkey and AltO to reset its origin.



Orientation tweaks

You can now change the orientation of the monkey by either rotating it (either in Edit mode or Object mode) or by changing the Tracking Axes under Animation Hacks (with the monkey selected). The arrangement of monkeys can, of course, be further enhanced by editing the curve.

To transform all monkeys into real objects, first Ctrl⇧ ShiftA to Make Duplicates Real. All monkeys are now real objects, but still linked copies. To change this, Object→Make Single User→Object&Data then choose All.

Note
There are many alternatives to Dupliframes. Which tool to use depends on context.

1. To use a small curve as a profile and a larger curve as a path, simply use the former as a Bevel Object to the latter.
2. To arrange objects along a curve, combining an Array Modifier and a Curve Modifier is often useful.
3. Dupliverts can be used to arrange objects, for example, along a circle or across a subdivided plane.

External links

- Blender Artists: *Dupliframes in 2.5*

Tracking

Mode: Object mode

Panel: Object » Constraints

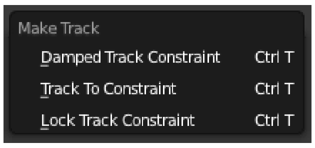
Hotkey: CtrlT

Menu: Object » Track » Make Track

Description

Tracking consists of one object watching another. The watcher is the “Tracker” and the watched is the “Target”. If the target moves the tracker rotates; if the tracker moves the tracker rotates. In both cases the tracker maintains a constant heading towards the target.

Types of Tracking


Make Track
Damped Track Constraint Ctrl T
Track To Constraint Ctrl T
Lock Track Constraint Ctrl T
Make Track menu.

To make one or more objects track another object (the target), select at least two objects and press CtrlT. The active object becomes the target and the other objects the trackers. The (*Make Track menu*) provides several options for creating the initial tracking:

Track To Constraint

The Track To constraint applies rotations to its owner, so that it always points a given “To” axis towards its target, with another “Up” axis permanently maintained as much aligned with the global Z axis (by default) as possible. See: Track To Constraint

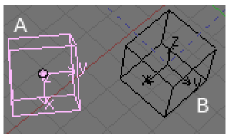
Locked Track Constraint

The Locked Track constraint is a bit tricky to explain, both graphically and textually. Basically, it is a Track To Constraint, but with a locked axis, i.e. an axis that cannot rotate (change its orientation). Hence, the owner can only track its target by rotating around this axis, and unless the target is in the plane perpendicular to the locked axis, and crossing the owner, this owner cannot really point at its target. See: Locked Track Constraint

Damped Track Constraint

The Damped Track constraint constrains one local axis of the owner to always point towards Target. See Damped Track Constraint

Old Track

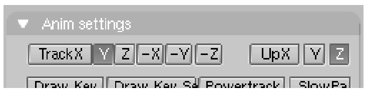


Old Track “constraint”.

This is an older algorithm prior to version 2.30, and is similar to Track To constraint in that no axis is locked. This algorithm merely tries to keep a “To” axis pointed at the target. The tracking object will usually end up in an odd orientation when this constraint is first applied. In order to get correct results use AltR when applying or changing the tracking or “Up” axis. However, the preferred method to use is Track To constraint.

Let’s assume you have nonetheless selected Old Track in the dialog with two cubes selected; see (*Old Track “constraint”*). By default the inactive object(s) track the active object so that their local +Y axis points to the tracked object. Cube “`A`” is tracking cube “`B`” using the Old Track constraint. You can see that “`A`”’s +Y axis is pointing at “`B`” but at an odd orientation. This typically happens if the object already has a rotation of its own. You can produce correct tracking by canceling the rotation on the tracking object using AltR.

The orientation of the tracking object is also set such that the chosen “Up” axis is pointing upward.


Anim settings
TrackX Y Z -X -Y -Z UpX Y Z
Draw Key Draw Key Se Powertrack SlowPa
Setting track axis.

If you want to change this you need to get to the Anim settings panel where Old Track’s settings are accessed. First select the tracking object (not the target) and change the Button window to Object context by clicking the icon ( ), or F7; see (*Setting track axis*).

You then have the option of selecting the *Tracking axis* from the first column-set of six radio buttons and/or selecting the *upward-pointing* axis from the second column-set in the Anim Setting panel. Each time you change the “Up” axis you need to apply AltR otherwise the tracking object will continue to track with the old orientation. This is one of the drawbacks to using Old Track.

To clear or remove an old track “constraint”, select the tracking object and press AltT. As with clearing a parent constraint, you must choose whether to lose or save the rotation imposed by the tracking.

Note
AltT command works (and is useful) only for the Old Track “constraint”. To clear the Track To and Locked Track constraints, just delete them directly from the stack at the Constraints panel.

Hints

The *active* object always becomes the target object to be tracked. In all but Old Track a blue dashed line is drawn between the tracker and target indicating that a tracking constraint is in place between the corresponding objects. If you see an object tracking another object without a dashed blue line then you know the tracking object is using the Old Track “constraint”.

Invalid Tracking or settings

If you choose an invalid tracking “To” and/or “Up” axis, the tracking object keeps it current orientation and ignores the incorrect selections. For example, if you choose the +Z axis as the “To” axis and also choose the +Z axis as the “Up” axis, you have chosen an invalid combination because you can’t have the tracking object’s +Z axis doing two different things at the same time.

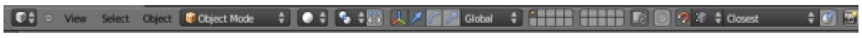
If you have problems setting up the correct “To” and “Up” axes you may want to turn on the tracking object’s local axes. You can do this from the Draw panel by clicking on the Axis button. See The Interface chapter for further details on the Draw panel.

Edit Mode

Entering Edit Mode

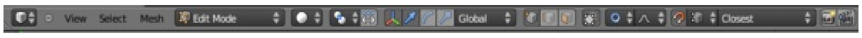
You can work with geometric objects in two modes.

Object mode
: Operations in Object Mode affect the whole object.
Object mode has the following header in the 3D view:



Object Mode header

Edit mode
: Operations in Edit mode affect only the geometry of an object, but not global properties such as location or rotation.
Edit mode has the following header in the 3D view:



Edit Mode header

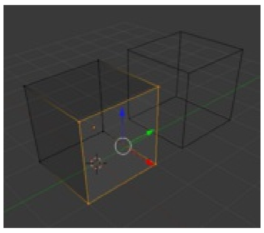
Tools and modes in the 3D view header are (left to right):

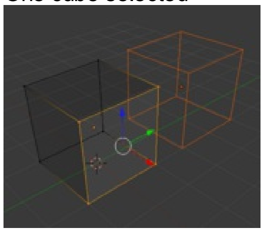
- View, Select, and Mesh menus
- Blender Mode
- Display method for 3D view
- Pivot center
- 3D manipulator widget
- Selection mode
- Depth buffer clipping (hide
- Proportional editing
- Snap
- OpenGL render

You can switch between the Object and Edit modes with the ⇆ Tab key. You can change to any mode by selecting the desired Mode in the menu in the 3d view header.

After creating an object you may be immediately placed in Edit mode – depending on whether the Switch to Edit Mode button is toggled in the User Preferences Editing tab. Edit mode only applies to one object at a time, the *active*, or most recently selected, object.

Visualization


One cube selected


Two cubes selected
before entering Edit mode

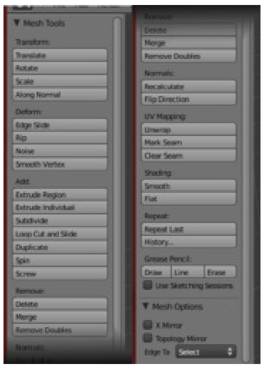
By default, Blender highlights selected geometry in orange in both Object mode and Edit mode. The color can be changed in the User Preferences (CtrlAltU→Themes.)

In Object mode with Wireframe shading enabled (Z), objects are displayed in black when unselected and in orange when selected. If more than one object is selected, all selected object except the active object, typically the object last selected, is displayed in a darker orange color. Similarly, in Edit mode, unselected geometry is drawn in black while selected faces, edges, or vertices are drawn in orange. The active face is highlighted in white.

In Edit mode, only one mesh can be edited at the time. However, several objects can be joined into a single mesh (CtrlJ in Object mode) and then separated again (P in Edit mode). If multiple objects are selected before entering Edit mode, all the selected objects remain highlighted in orange indicating that they are part of the active selection set.

If two vertices joined by an edge are selected in Vertex selection mode, the edge between them is highlighted too. Similarly, if enough vertices or edges are selected to define a face, that face is also highlighted.

Tool Shelf



The Tool Shelf panel in edit mode (panel split in two parts for layout reasons)

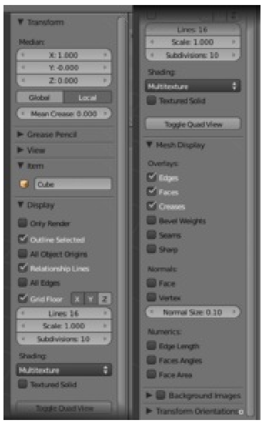
Open/close the Mesh Tools panel using T. When entering Edit mode, several mesh tools become available.

Most of these tools are also available as shortcuts (displayed in the Tooltips for each tool) and/or in the Specials menu (W), the Edge menu (CtrlE) ,and Face menu (CtrlF). For each tool a context-dependent menu is opened at the bottom of the Tool Shelf.

Even more mesh editing tools can be enabled in the User Preferences' Add-Ons section. The development of new tools is regularly announced on Blender-related sites and forums.

For further information on panels see the Reference panels section.

Properties Shelf



The Properties Shelf panel in edit mode (panel split in two parts for layout reasons)

Open/close the Properties Shelf using N.

In the Properties Shelf, panels directly related to mesh editing are the Transform panel, where numeric values can be entered, and the Mesh Display panel, where for example normals and numeric values for distances, angles, and areas can be turned on.

Other useful tools are found in the Properties Editor under the Object's and Object Data's Context buttons, including display options and Vertex groups.

For further information on panels see the Reference panels section.

Mesh Display

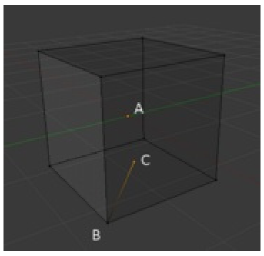
TODO...

- Overlays
- Normals
- Edge/Face Info

Vertices, Edges and Faces

In basic meshes, everything is built from three basic structures: *Vertices*, *Edges* and *Faces* (we're not talking about curves, NURBS, and so forth here). But there is no need to be disappointed: this simplicity still provides us with a wealth of possibilities that will be the foundation for all our models.

Vertices



Vertex example

A vertex is primarily a single point or position in 3D space. It is usually invisible in rendering and in Object mode. Don't mistake the center point of an object for a vertex. It looks similar, but it's bigger and you can't select it. (*Vertex example*) shows the center point labeled as "`A`". "`B`" and "`C`" are vertices.

A simple way to create a new vertex is to click Ctrl LMB in Edit mode. Of course, as a computer screen is two-dimensional, Blender can't determine all three vertex coordinates from a single mouse click, so the new vertex is placed at the depth of the 3D cursor. Using the method described above, any vertices selected previously are automatically connected to the new ones by an edge. In the image above, the vertex labeled C is a new vertex added to the cube with a new edge added between in B and C.

Edges

An edge always connects two vertices by a straight line. The edges are the "wires" you see when you look at a mesh in wireframe view. They are usually invisible on the rendered image. They are used to construct faces. Create an edge by selecting two vertices and pressing F.

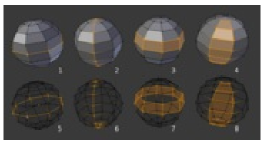
Faces

A face is the highest level structure in a mesh. Faces are used to build the actual surface of the object. They are what you see when you render the mesh. A face is defined as the area between either three (triangles) or four (quadrangles) vertices, with an edge on every side. Triangles are always flat and therefore easy to calculate. On the other hand, quadrangles "deform well" and are therefore preferred for subdivision modeling.

Take care when using four-sided faces (quads), because internally they are simply divided into two triangles each. Four-sided faces only work well if the face is pretty much flat (all points lie within one imaginary plane) and convex (the angle at no corner is greater than or equal to 180 degrees). This is the case with the faces of a cube, for example. That's why you can't see any diagonal in its wireframe model, because they would divide each square face into two triangles.

While you could build a cube with triangular faces, it would just look more confusing in Edit mode. An area between three or four vertices, outlined by edges, doesn't have to be a face. If this area does not contain a face, it will simply be transparent or non-existent in the rendered image. To create a face, select three or four suitable vertices and press F.

Loops



Edge and Face Loops

Edge and Face Loops are sets of faces or edges that form continuous "loops" as shown in (*Edge and Face Loops*). The top row (`1`-`4`) shows a solid view, the bottom row (`5`-`8`) a wireframe view of the same loops.

Note that loops `2` and `4` do not go around the whole model. Loops stop at so called poles because there is no unique way to continue a loop from a pole. Poles are vertices that are connected to either three, five, or more edges. Accordingly, vertices connected to exactly one, two or four edges are not poles.

In the image above, loops that do not end in poles are cyclic (`1` and `3`). They start and end at the same vertex and divide the model into two partitions. Loops can be a quick and powerful tool to work with specific, continuous regions of a mesh and are a prerequisite for organic character animation. For a detailed description of how to work with loops in Blender, please refer to the Manual page on Edge and Face Tools.

Edge Loops

Loops `1` and `2` in (*Edge and Face Loops*) are edge Loops. They connect vertices so that each one on the loop has exactly two neighbors that are not on the loop and placed on both sides of the loop (except the start and end vertices in the case of poles).

Edge Loops are an important concept especially in organic (subsurface) modeling and character animation. When used correctly, they allow you to build models with relatively few vertices that look very natural when used as subdivision surfaces and deform very well in animation.

Take (*Edge Loops in organic modeling*) as an example: the edge loops follow the natural contours and deformation lines of the skin and the underlying muscles and are more dense in areas that deform more when the character moves, for example at the shoulders or knees.

Further details on working with Edge Loops can be found in Edge Loop Selection.

Face Loops

These are a logical extension of Edge Loops in that they consist of the faces between two Edge Loops, as shown in loops `3` and `4` in (*Edge and Face Loops*). Note that for non-circular loops (`4`) the faces containing the poles are not included in a Face Loop.

Further details on working with Face Loops can be found in Face Loop Selection.

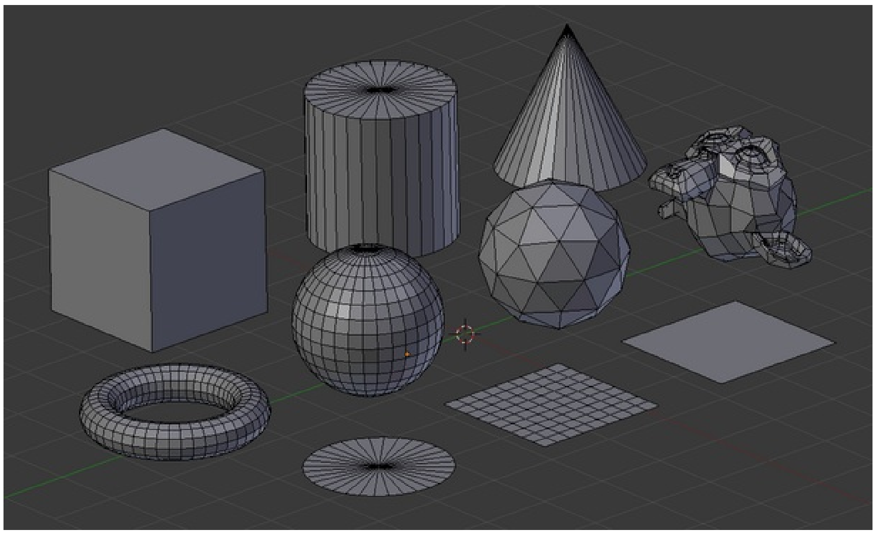
Mesh Primitives

Mode: Object mode

Hotkey: ⇧ ShiftA

Menu: Add » Mesh

A common object type used in a 3D scene is a mesh. Blender comes with a number of “primitive” mesh shapes that you can start modeling from.


Blender's ten standard primitives

Options included in more than one primitive are:

Radius
: Sets the starting size for Circle, Cylinder, Cone, UVSphere and IcoSphere.

Depth
: Sets the starting length for Cylinder and Cone.

Note about planar primitives
You can make a planar mesh three-dimensional by moving one or more of the vertices out of its plane (applies to Plane, Circle and Grid). A simple circle is actually often used as a starting point to create even the most complex of meshes.

Plane

A standard plane contains four vertices, four edges, and one face. It is like a piece of paper lying on a table; it is not a real three-dimensional object because it is flat and has no thickness. Objects that can be created with planes include floors, tabletops, or mirrors.

Cube

A standard cube contains eight vertices, twelve edges, and six faces, and is a real three-dimensional object. Objects that can be created out of cubes include dice, boxes, or crates.

Circle

A standard circle is comprised of *n* vertices. The number of vertices and radius can be specified in the context panel in the Tool Shelf which appears when the circle is created.

Vertices
: The number of vertices that define the circle. The more vertices the circle contains, the smoother its contour will be; see (*“Circles” obtained with various settings*). In contrast, a circle with only 3 vertices is actually a triangle — the circle is actually the standard way of adding polygons such as triangles, pentagons, et cetera.

Radius
: Sets the radius of the circle.

Fill Type
: Set how the circle will be filled

  Triangle Fan
  : Fill with triangular faces which share a vertex in the middle.

  Ngon
  : fill with a single ngon

Nothing
: Do not fill. Creates only the outer ring of vertices

UV Sphere

A standard UV sphere is made out of *n* segments and *m* rings. The level of detail and radius can be specified in the context panel in the Tool Shelf which appears when the UV sphere is created. Increasing the number of segments and rings makes the surface of the UV sphere smoother.

Segments
: Number of vertical segments. Like Earth’s meridians, going pole to pole and

Rings
: Number of horizontal segments. These are like Earth’s parallels.

Note
If you specify a six segment, six ring UVsphere you’ll get something which, in top view, is a hexagon (six segments), with five rings plus two points at the poles. Thus, one ring fewer than expected, or one more, if you count the poles as rings of radius 0.

Icosphere

An icosphere is a polyhedra sphere made up of triangles. The number of subdivisions and radius can be specified in the context panel in the Tool Shelf after the Icosphere is created. Icospheres are normally used to achieve a more isotropical and economical layout of vertices than a UV sphere.

Subdivisions
: How many recursions are used to define the sphere. Increasing the number of subdivisions makes the surface of the Icosphere smoother. At level 1 the Icosphere is an icosahedron, a solid with 20 equilateral triangular faces. Any increasing level of subdivision splits each triangular face into four triangles, resulting in a more spherical appearance.

Size
: The radius of the sphere.

Note
It is possible to add an icosphere subdivided 500 times. Adding such a dense mesh is a sure way to cause a program crash. An icosphere subdivided 10 times would have 5,242,880 triangles, so be very careful about this!

Cylinder

A standard cylinder is made out of *n* vertices. The number of vertices in the circular cross-section can be specified in the context panel in the Tool Shelf that appears when the object is created; the higher the number of vertices, the smoother the circular cross-section becomes. Objects that can be created out of cylinders include handles or rods.

Vertices
: Then number of vertical edge loops used to define the cylinder.

Radius
: Sets the radius of the cylinder.

Depth
: Sets the height of the cylinder.

Cap Fill Type
: Similar to circle (see above). When set to none, the created object will be a tube. Objects that can be created out of tubes include pipes or drinking glasses (the basic difference between a cylinder and a tube is that the former has closed ends).

Cone

A standard cone is made out of *n* vertices. The number of vertices in the circular base, dimensions and option to close the base of the cone can be specified in the context panel in the Tool Shelf that appears when the object is created; the higher the number of vertices, the smoother the circular base becomes. Objects that can be created out of cones include spikes or pointed hats.

Vertices
: The number of vertical edge loops used to define the cone.

Radius 1
: Sets the radius of the base of the cone.

Radius 2
: Sets the radius of the tip of the cone. A value of 0 will produce a standard cone shape.

Depth
: Sets the height of the cylinder.

Base Fill Type
: Similar to circle (see above).

Torus

A doughnut-shaped primitive created by rotating a circle around an axis. The overall dimensions are defined by the Major and Minor Radius. The number of vertices (in segments) can be different for the circles and is specified in the context panel in the Tool Shelf with both radii (Major Segments and Minor Segments).

Major Radius
: Radius from the origin to the center of the cross sections

Minor Radius
: Radius of the torus's cross section

Major Segments
: Number of segments for the main ring of the torus. If you think of a torus as a "spin" operation around an axis, this is how many steps in the spin.

Minor segments
: Number of segments for the minor ring of the torus. This is the number of vertices of each circular segment.

Use Int+Ext Controls
: Change the way the torus is defined:

Exterior Radius
: When Use Int+Ext Controls is active, if viewed along the major axis, this is the radius from the center to the outer edge.

Interior Radius
: When Use Int+Ext Controls is active, if viewed along the major axis, this is the radius of the hole in the center.

Grid

A standard grid is made out of *n* by *m* vertices. The resolution of the x-axis and y-axis can be specified in the context panel in the Tool Shelf which appears when the object is created; the higher the resolution, the more vertices are created. Example objects that can be created out of grids include landscapes (with the proportional editing tool or Displace modifier) and other organic surfaces. You can also obtain a grid when you create a plane and then use a subdivide modifier in Edit mode. However, there is a Landscape add-on available in the User Preferences.

X Subdivisions
: The number of spans in the x direction. Minimum of 3, creating two face loops.

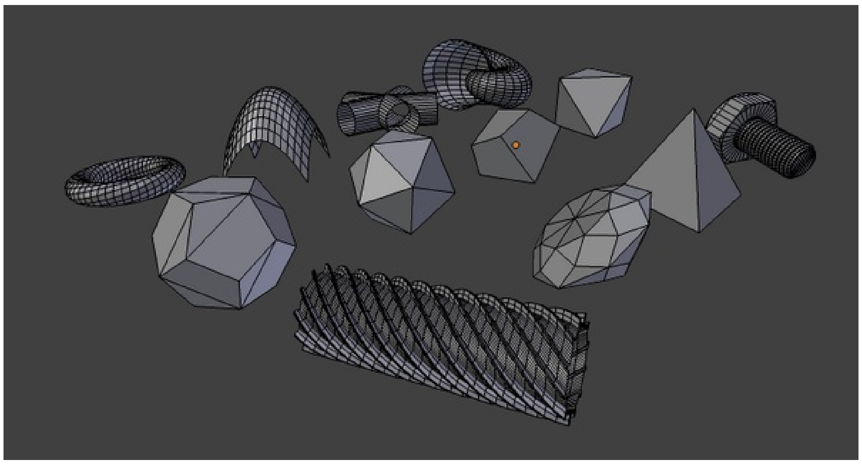
Y Subdivisions
: The number of spans in the y direction.

Size
: The length of the sides of the grid.

Monkey

This is a gift from old NaN to the community and is seen as a programmer's joke or “Easter Egg”. It creates a monkey's head once you press the Monkey button. The Monkey's name is “Suzanne” and is Blender's mascot. Suzanne is very useful as a standard test mesh, much like the Utah Tea Pot or the Stanford Bunny.

Add-ons


A few of the mesh primitives available as add-ons.

In addition to the basic geometric primitives, Blender has a constantly increasing number of script generated meshes to offer as pre-installed add-ons. These become available when enabled in the User Preferences' Add-ons section (filter by Add Mesh). Only a few are mentioned here:

Landscape
: Adds a landscape primitive. Many parameters and filters appear in the Tool Shelf.

Pipe Joints
: Adds one of five different pipe joint primitives. Radius, angle, and other parameters can be changed in the Tool Shelf.

Gears
: Adds a gear or a worm with many parameters to control the shape in the Tool Shelf.

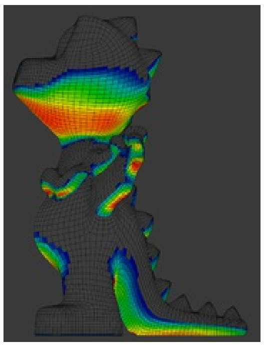
Mesh Analysis

Mesh analysis is useful for displaying attributes of the mesh that may impact certain use cases.

The mesh analysis works in editmode and shows areas with a high value in red, and areas with a low value in blue. Geometry outside the range is displayed grey.

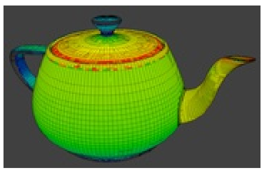
Currently the different modes target 3d-printing as their primary use.

Overhang


Overhang

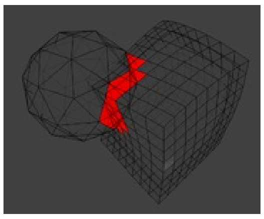
Extrusion 3D printers have a physical limit to the overhang that can be printed, this display mode shows the overhang with angle range and axis selection.

Thickness


Thickness

Printers have a limited *wall-thickness* where very thin areas can't be printed, this test uses ray casting and a distance range to the thickness of the geometry.

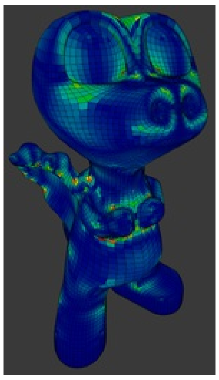
Intersections


Intersecting faces

Another common cause of problems for printing are intersections between surfaces, where the inside/outside of a model can't be reliably detected.

Unlike other display modes, intersections have no variance and are either on or off.

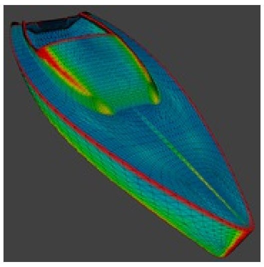
Distortion


Distorted Faces

Distorted geometry can cause problems since the triangulation of a distorted ngon is undefined.

Distortion is measured by faces which are not flat, with parts of the face pointing in different directions.

Sharp Edges


Sharp edges

Similar to wall-thickness, sharp edges can form shapes that are too thin to be able to print.

Warning

There are some known limitations with mesh analysis

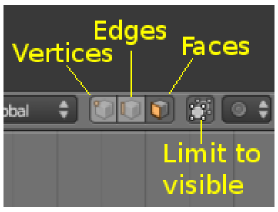
- Currently only displayed with deform modifiers.
- For high-poly meshes is slow to use while editing the mesh.

Selecting Mesh Components

There are many ways to select elements, and it depends on what Mesh Select Mode you are in as to what selection tools are available. First we will go through these modes and after that a look is taken at basic selection tools.

Selection Modes

Select Mode Header Widgets



Edit mode selection buttons

In Edit mode there are three different selection modes. You can enter the different modes by selecting one of the three buttons in the toolbar.

Using the buttons you can also use more than one selection mode at a time by ⇧ Shift LMB clicking the buttons.

Vertices
: Selected vertices are drawn in orange, unselected vertices in black, and the active or last selected vertex in white.

Edges
: In this mode the vertices are not drawn. Instead the selected edges are drawn in orange, unselected edges black, and the active or last selected edge in white.

Faces
: In this mode the faces are drawn with a selection point in the middle which is used for selecting a face. Selected faces and their selection point are drawn in orange, unselected faces are drawn in black, and the active or last selected face is highlighted in white.

Almost all modification tools are available in all three mesh selection modes. So you can Rotate, Scale, Extrude, etc. in all modes. Of course rotating and scaling a *single* vertex will not do anything useful without setting the pivot point to another location, so some tools are more or less applicable in some modes.

Note
The three selection mode buttons are only visible in Edit mode. The colors of selected, unselected and active components depend entirely on the current theme. Black, orange and white are from the default theme.

Select Mode Pop-up

Mode: Edit mode

Hotkey: Ctrl⇆ Tab

Mesh Select Mode menu

You can also choose a selection mode with the pop-up menu

Select Mode » Vertices
: Press Ctrl⇆ Tab and select Vertices from the pop-up menu, or press Ctrl⇆ Tab1.

Select Mode » Edges
: Press Ctrl⇆ Tab and select Edges from the pop-up menu, or press Ctrl⇆ Tab2.

Select Mode » Faces
: Press Ctrl⇆ Tab and select Faces from the pop-up menu, or press Ctrl⇆ Tab3.

Switching select mode

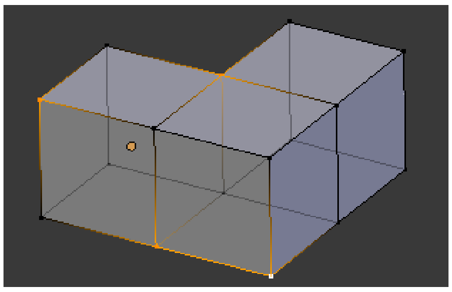
When switching modes in an “ascendant” way (i.e. from simpler to more complex), from Vertices to Edges and from Edges to Faces, the selected parts will still be selected if they form a complete element in the new mode.

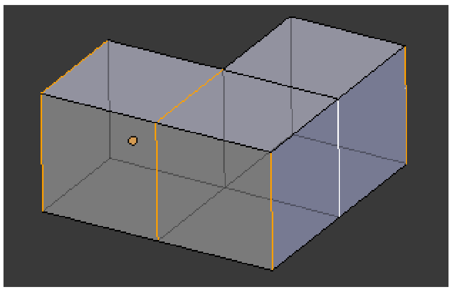
For example, if all four edges in a face are selected, switching from Edges mode to Faces mode will keep the face selected. All selected parts that do not form a complete set in the new mode will be unselected.

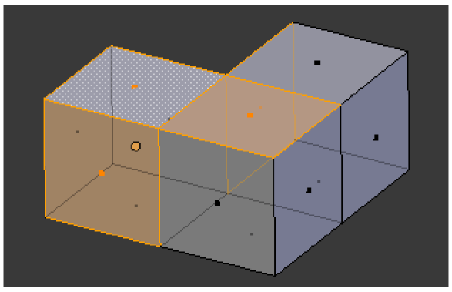
Hence, switching in a “descendant” way (i.e. from more complex to simpler), all elements defining the “high-level” element (like a face) will be selected (the four vertices or edges of a quadrangle, for example).

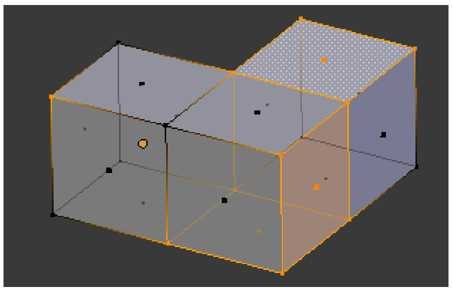
By holding Ctrl when selecting a higher selection mode, all elements touching the current selection will be added, even if the selection does not form a complete higher element.

See (*Vertices mode example*), (*Edges mode example*), (*Faces mode example*) and (*Mixed mode example*) for examples of the different modes.


Vertices mode example.


Edges mode example.


Faces mode example.


Mixed mode example.

Selection Tools

The select menu in edit mode contains tools for selecting components. These are described in more detail in the following pages.

Border Select
: Enables a rectangular region for selection

Circle Select
: Enables a circular shaped region for selection

(De)select All A
: Select all or none of the mesh components.

Invert Selection CtrlI
: Selects all components that are not selected, and deselect currently selected components.

Select Random
: Selects a random group of vertices, edges, or faces, based on a percentage value.

Checker Deselect
: Deselect alternating faces, to create a checker like pattern.

Select Sharp Edges
: This option will select all edges that are between two faces, forming an angle less than the user-specified value. The lower this angle limit, the sharper the selected edges will be. At **180°**, **all** “manifold” (see below) edges will be selected.

Linked Flat Faces (Ctrl⇧ ShiftAltF)
: Select connected faces based on a threshold of the angle between them. This is useful for selecting faces that are planar.

Interior Faces
: Select faces where all edges have more than 2 faces.

Side of Active
: Selects all data on the mesh in a single axis

Select Faces by Sides
: Selects all faces that have a specified number of edges.

Select Non Manifold (Ctrl⇧ ShiftAltM)
: Selects vertices that are not completely bound by geometry, including border edges, floating edges, and orphan vertices. Only available in Vertex mode.

Loose
: Select all vertices or edges that do not form part of a face.

Similar
: Select components based on how similar certain properties are to it.

More CtrlNum+
: Propagates selection by adding components that are adjacent to selected elements.

Less CtrlNum-
: Deselects components that form the bounds of the current selection

Mirror
: Select mesh items at the mirrored location.

Linked CtrlL
: Selects all components that are connected to the current selection.

Shortest Path
: Selects the shortest vertex path between two selected vertices or edges

Edge Loop
: Selects a loop of edges from a selected edge

Edge Ring
: Selects edges parallel to a selected edge in the same ring of faces

Loop Inner-Region
: Converts a closed selection of edges to the region of faces it encloses

Boundary Loop
: Converts a selection of faces to the ring of edges enclosing it

Selectable Elements

As we have seen in the mesh structure page, meshes are made of different element types (even though they are all inter-related: in a way, they are different "views", "representations", of the same basic data…), "vertices", "edges" and "faces".

Hence, you can select different parts of a mesh using one of these three types. There is one key point to understand here: *when you select a type of element (e.g. some edges), you* ***implicitly*** *select the other types of corresponding elements (e.g. all vertices defining those edges, as well as faces fully defined by these same edges)*. This is very important, as some tools only work on vertices, edges and/or faces: if you use a "face" tool with a selection of vertices, only the faces defined by these vertices will be affected.

In general, you will only select one type of element at a time, depending on the "select mode" you are using. However, you can successively add different elements to a same selection, switching between these select modes (see below for what is selected after switching select mode), or even use a "combined" select mode, also described below.

Select Modes

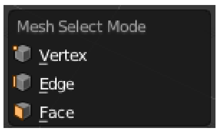
You have two ways to switch between select modes:

Select Mode popup

Mode: Edit mode

Hotkey: Ctrl⇆ Tab

In Edit mode there are three different select modes for meshes; see (*Select Mode menu*).



Select Mode menu.

Select Mode » Vertices
: Press Ctrl⇆ Tab and select Vertices from the popup menu, or press Ctrl⇆ Tab1 NumPad. The selected vertices are drawn in yellow and unselected vertices are drawn in a pink colour.

Select Mode » Edges
: Press Ctrl⇆ Tab and select Edges from the popup menu, or press Ctrl⇆ Tab2 NumPad. In this mode the vertices are not drawn. Instead the selected edges are drawn in yellow and unselected edges are drawn in a black colour.

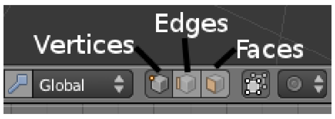
Select Mode » Faces
: Press Ctrl⇆ Tab and select Faces from the popup menu, or press Ctrl⇆ Tab3 NumPad. In this mode the faces are drawn with a selection point in the middle which is used for selecting a face. Selected faces are drawn in yellow with the selection point in orange, unselected faces are drawn in black.

Almost all modification tools are available in all three modes. So you can Rotate, Scale, Extrude, etc. in all modes. Of course rotating and scaling a *single* vertex will not do anything useful, so some tools are more or less applicable in some modes.

Select Mode header widgets

Mode: Edit mode

Panel: Header of the 3D View



Edit mode select mode buttons.

You can also enter the different modes by selecting one of the three buttons in the toolbar; see (*Edit mode select buttons*).

Using the buttons you can also enter "**mixed**" or "combined" mode by ⇧ Shift LMB clicking the buttons. This will allow you to select vertices, edges and/or faces at the same time!

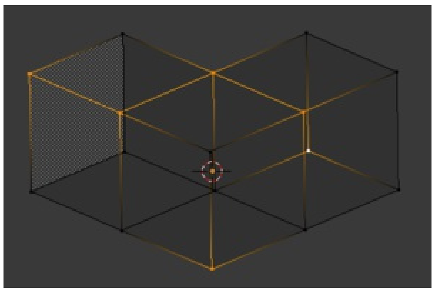
Note
The "Mode Selection" buttons are only visible for meshes in Edit mode.

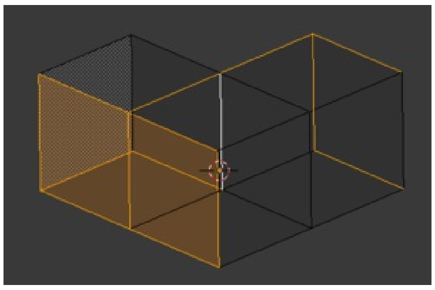
Selected elements after switching select mode

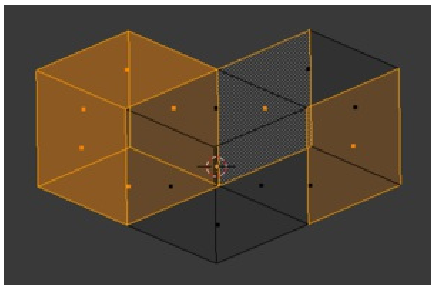
When switching modes in an "ascendant" way (i.e. from simpler to more complex), from Vertices to Edges and from Edges to Faces, the selected parts will still be selected if they form a complete set in the new mode. For example, if all four edges in a face are selected, switching from Edges mode to Faces mode will keep the face selected. All selected parts that do not form a complete set in the new mode will be unselected.

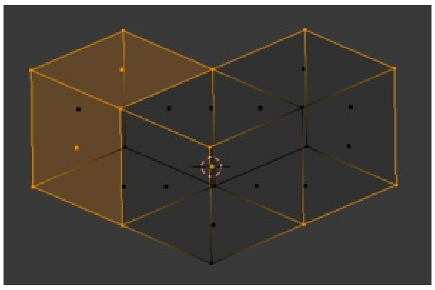
Hence, switching in a "descendant" way (i.e. from more complex to simpler), all elements defining the "high-level" element (like a face) will be selected (the four vertices or edges of a quadrangle, for example).

See (*Vertices mode example*), (*Edges mode example*), (*Faces mode example*) and (*Mixed mode example*) for examples of the different modes.


none Vertices mode example.


Edges mode example.


Faces mode example.


Mixed mode example.

Basic Selection

Mode: Edit mode

Hotkey: RMB and ⇧ Shift RMB

The most common way to select an element is to RMB on that item; this will replace the existing selection with the new item.

Adding to a Selection

To add to the existing selection, hold down ⇧ Shift while right clicking. Clicking again on a selected item will deselect it.

As in Object mode, there is a unique *active* element, displayed in a lighter shade (in general, the last element selected). Depending on the tools used, this element might be very important!

Note that there is no option to choose what element to select between overlapping ones (like the Alt RMB click in Object mode). However, if you are in solid, shaded, or textured viewport shading mode (not bounding box or wireframe), you will have a fourth button in the header that looks like a cube, just right of the select mode ones.

When enabled, this limits your ability to select based on visible elements (as if the object was solid), and prevents you from accidentally selecting, moving, deleting or otherwise working on backside or hidden items.

Selecting Elements in a Region

Mode: Edit mode

Hotkey: B, C, and Ctrl LMB click and drag

Region selection allows you to select groups of elements within a 2D region in your 3D view. The region can be either a circle or rectangle. The rectangular region, or "*Border Select*", and the circular region are both available in Edit mode and Object mode.

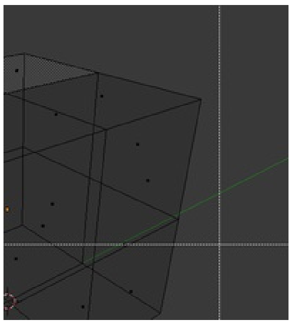
Note
What is selected using both these tools is affected by the Limit Selection to visible feature (available under the 3D viewport) in Solid Viewport Shading Mode.

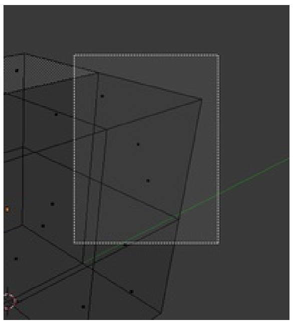
For example,

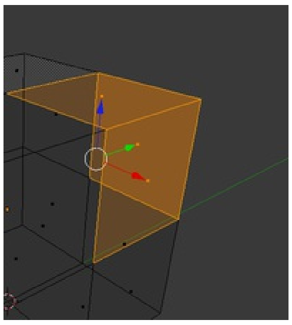
1. in solid shading mode and face selection mode, all faces *within* the selection area will be selected;
2. whilst in the wireframe shading mode and face selection mode, only faces whose handle are within the selection area will be selected.

Rectangular region (Border select)

Border Select is available in either Edit mode or Object mode. To activate the tool use the B. Use Border Select to select a group of objects by drawing a rectangle while holding down LMB. In doing this you will select all objects that lie within or touch this rectangle. If any object that was last active appears in the group it will become selected *and* active.


Start


Selecting


Complete

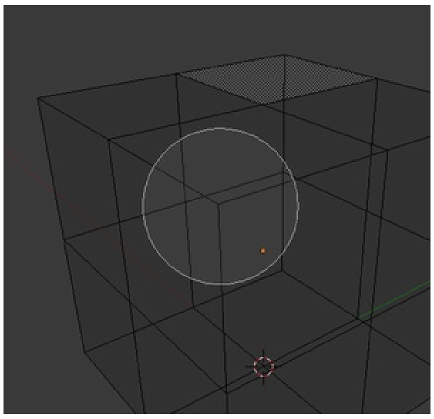
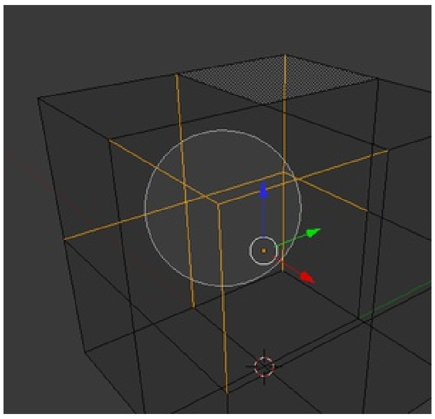
In (*Start*), Border Select has been activated and is indicated by showing a dotted cross-hair cursor. In (*Selecting*), the *selection region* is being chosen by drawing a rectangle with the LMB. The selection area is only covering the selection handles of three faces. Finally, by releasing LMB the selection is complete; see (*Complete*).

Note
Border select adds to the previous selection, so in order to select only the contents of the rectangle, deselect all with A first. In addition, you can use MMB while you draw the border to deselect all objects within the rectangle.

Circular region

This selection tool is available both in Edit mode and Object mode and can be activated with C. Once activated, the cursor changes to a dashed cross-hair with a 2D circle surrounding it. The tool will operate on whatever the current select mode is. Clicking or dragging with the LMB, causing elements to be inside the circle will cause those elements to be selected.

You can enlarge or shrink the circle region using + NumPad and - NumPad, or the Wheel.



Before After
Circle Region Select

(*Circle Region Select*) is an example of selecting edges while in Edge Select Mode. As soon as an edge intersects the circle the edge becomes selected. The tool is interactive such that edges are selected while the circle region is being dragged with the LMB.

If you want to deselect elements, either hold MMB or Alt LMB and begin clicking or dragging again.

For Faces select mode, the circle must intersect the face indicators usually represented by small pixel squares; one at the center of each face.

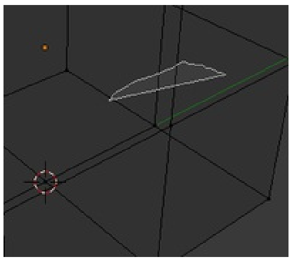
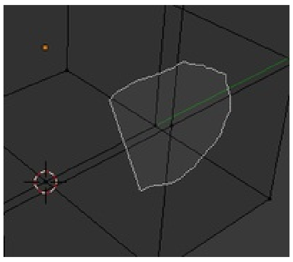
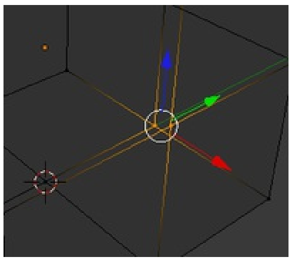
To exit from this tool, click RMB, or hit the Esc key.

Lasso region

Lasso select is similar to Border select in that you select objects based on a region, except Lasso is a hand-drawn region that generally forms a circular/round-shaped form; kind of like a lasso.

Lasso is available in either Edit Mode or Object Mode. To activate the tool use the Ctrl LMB while dragging. The one difference between Lasso and Border select is that in Object mode, Lasso only selects objects where the lasso region intersects the objects' center.

To deselect, use Ctrl⇧ Shift LMB while dragging.




Start Selecting Complete
Lasso selection

(*Lasso selection*) is an example of using the Lasso select tool in Vertex Select Mode.

Additional Selection Tools

The select menu in edit mode contains additional tool for selecting components:

(De)select All A
    Select all or none of the mesh components.
Invert Selection CtrlI
    Selects all components that are not selected, and deselect currently selected components.
More CtrlNum+
    Propagates selection by adding components that are adjacent to selected elements.
Less CtrlNum-
    Deselects components that form the bounds of the current selection

Advanced Selection

The select menu in edit mode contains additional tool for selecting components:

Mirror
: Select mesh items at the mirrored location.

Linked CtrlL
: Selects all components that are connected to the current selection.

Select Random
: Selects a random group of vertices, edges, or faces, based on a percentage value.

Checker Deselect
: Deselect alternating faces, to create a checker like pattern.

Select Sharp Edges
: This option will select all edges that are between two faces forming an angle less than a given value, which is asked you *via* a small pop-up dialog. The lower is this angle limit, the sharper will be the selected edges. At **180°**, **all** “manifold” (see below) edges will be selected.

Linked Flat Faces (Ctrl⇧ ShiftAltF)
: Select connected faces based on a threshold of the angle between them. This is useful for selecting faces that are planar.

Select Non Manifold (Ctrl⇧ ShiftAltM)
: Selects vertices that are not completely bound by geometry, including border edges, floating edges, and orphan vertices. Only available in Vertex and Edge mode.

Interior Faces
: Select faces where all edges have more than 2 faces.

Side of Active
: Selects all data on the mesh on a single axis

Select Faces by Sides
: Selects all faces that have a specified number of edges.

Loose Geometry
: Select all vertices or edges that do not form part of a face.

Select Similar

Mode: Edit mode

Hotkey: ⇧ ShiftG

Menu: Select » Similar...

Select components that have similar attributes to the ones selected, based on a threshold that can be set in tool properties after activating the tool. Tool options adjust to the active selection mode

Vertex Selection Mode

Normal

Selects all vertices that have normals pointing in similar directions to those currently selected.

Amount of Adjacent Faces

Selects all vertices that have the same number of faces connected to them.

Vertex Groups

Selects all vertices in the same vertex group.

Amount of connecting edges

Selects all vertices that have the same number of edges connected to them.

Edge Selection Mode

Length

Selects all edges that have a similar length as those already selected.

Direction

Selects all edges that have a similar direction (angle) as those already selected.

Amount of Faces Around an Edge

Selects all edges that belong to the same number of faces.

Face Angles

Selects all edges that are between two faces forming a similar angle, as with those already selected.

Crease

Selects all edges that have a similar Crease value as those already selected. The Crease value is a setting used by the Subsurf Modifier.

Bevel

Selects all edges that have the same Bevel Weight as those already selected.

Seam

Selects all edges that have the same Seam state as those already selected. Seam is a true/false setting used in UV-texturing.

Sharpness

Selects all edges that have the same Sharp state as those already selected. Sharp is a true/false setting (a flag) used by the EdgeSplit Modifier.

Face Selection Mode

Material

Selects all faces that use the same material as those already selected.

Image

Selects all faces that use the same UV-texture as those already selected (see UV-texturing pages).

Area

Selects all faces that have a similar area as those already selected.

Polygon Sides

Selects all faces that have the same number of edges.

Perimeter

Selects all faces that have a similar perimeter as those already selected.

Normal

Selects all faces that have a similar normal as those selected. This is a way to select faces that have the same orientation (angle).

Co-planar

Selects all faces that are (nearly) in the same plane as those selected.

Selecting Loops

You can easily select loops of components:

Edge Loops and Vertex Loops

Mode: Edit mode → Vertex or Edge select mode

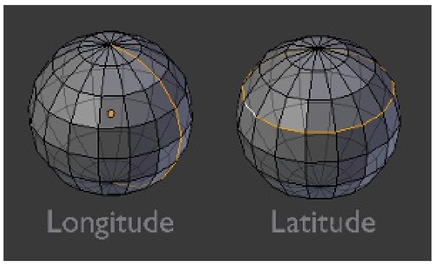
Hotkey: Alt RMB or CtrlE → Edge Loop

Menu: Select » Edge Loop or Mesh » Edges » Edge Loop

Holding Alt while selecting an edge selects a loop of edges that are connected in a line end to end, passing through the edge under the mouse pointer. Holding Alt⇧ Shift while clicking adds to the current selection.

Edge loops can also be selected based on an existing edge selection, using either Select » Edge Loop, or the Edge Loop Select option of the Edge Specials menu (CtrlE).

Vertex mode
In Vertex select mode, you can also select edge loops, by using the same hotkeys, *and clicking on the edges* (not on the vertices).



Longitudinal and latitudinal edge loops.

The left sphere shows an edge that was selected longitudinally. Notice how the loop is open. This is because the algorithm hit the vertices at the poles and terminated because the vertices at the pole connect to more than four edges. However, the right sphere shows an edge that was selected latitudinally and has formed a closed loop. This is because the algorithm hit the first edge that it started with.

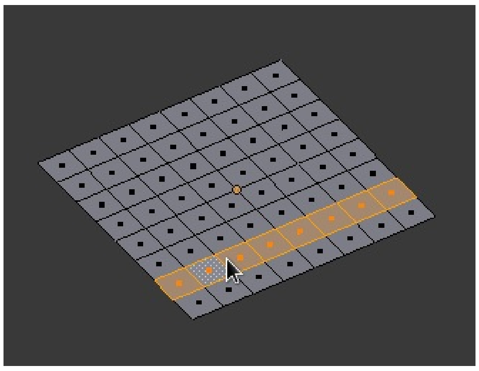
Face Loops

Mode: Edit mode → Face or Vertex select modes

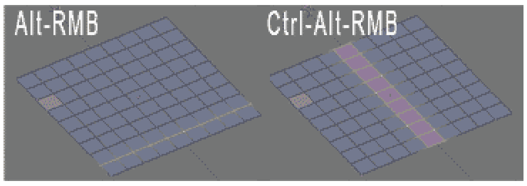
Hotkey: Alt RMB

In face select mode, holding Alt while selecting an **edge** selects a loop of faces that are connected in a line end to end, along their opposite edges.

In vertex select mode, the same can be accomplished by using CtrlAlt to select an edge, which selects the face loop implicitly.


Face loop selection.

This face loop was selected by clicking with Alt RMB on an edge, in face select mode. The loop extends perpendicular from the edge that was selected.



Alt versus CtrlAlt in vertex select mode.

A face loop can also be selected in Vertex select mode. Technically CtrlAlt RMB will select an Edge Ring, however in Vertex select mode, selecting an Edge Ring implicitly selects a Face Loop since selecting opposite edges of a face implicitly selects the entire face.

Edge Ring

Mode: Edit mode → Edge select mode

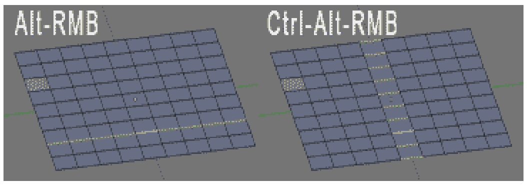
Hotkey: CtrlAlt RMB or CtrlE → Select » Edge Ring

Menu: Select » Edge Ring or Mesh » Edges » Edge Ring

In Edge select mode, holding CtrlAlt while selecting an edge selects a sequence of edges that are not connected, but on opposite sides to each other continuing along a face loop.

As with edge loops, you can also select edge rings based on current selection, using either Select » Edge Ring, or the Edge Ring Select option of the Edge Specials menu (CtrlE).

Vertex mode
In Vertex select mode, you can use the same hotkeys when *clicking on the edges* (not on the vertices), but this will directly select the corresponding face loop…



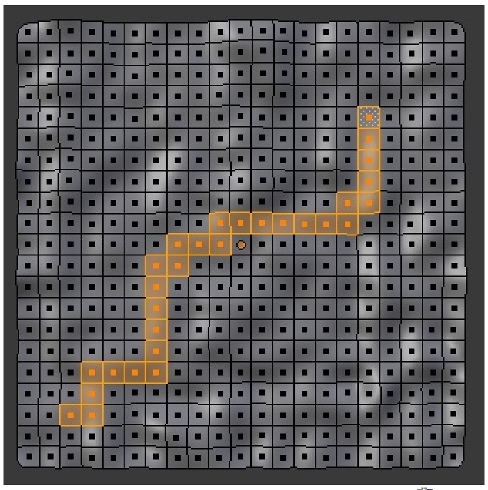
A selected edge loop, and a selected edge ring.

In (*A selected edge loop, and a selected edge ring*), the same edge was clicked on but two different “groups of edges” were selected, based on the different commands. One is based on edges during computation and the other is based on faces.

Path Selection

Mode: Edit mode

Hotkey: Ctrl RMB and the menu item Select → Shortest Path


Select a face or vertex path with Ctrl RMB

Selects all geometry along the shortest path from the active vertex/edge/face to the one which was selected.

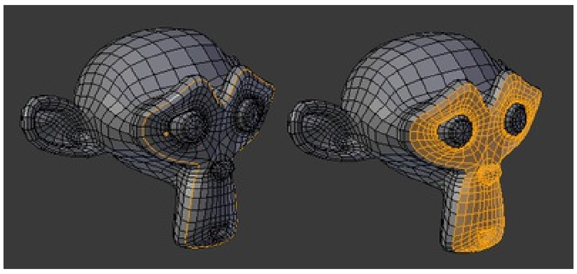
Loop Inner-Region

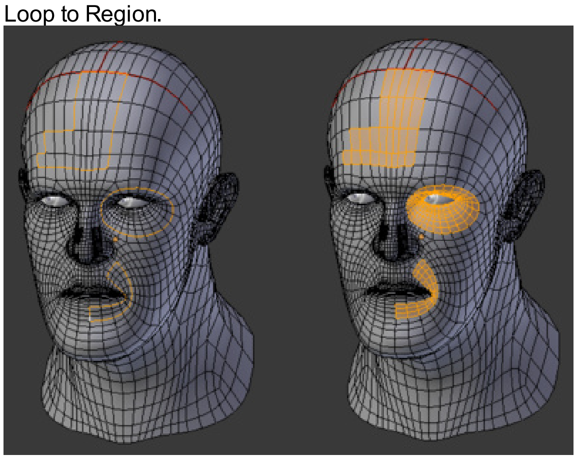
Mode: Edit mode → Edge select mode

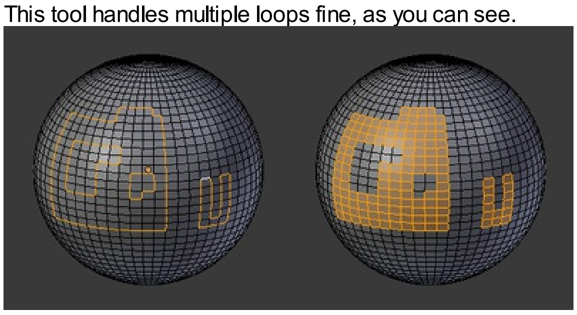
Hotkey: CtrlE → Select Loop Inner-Region

Menu: Select » Select Loop Inner-Region or Mesh » Edges » Select Loop Inner-Region

Select Loop Inner-Region selects all edges that are inside a closed loop of edges. While it is possible to use this operator in Vertex and Face selection modes, results may be unexpected. Note that if the selected loop of edges is not closed, then all connected edges on the mesh will be considered inside the loop.


Loop to Region.


This tool handles multiple loops fine, as you can see.


This tool handles “holes” just fine as well.

Boundary Loop

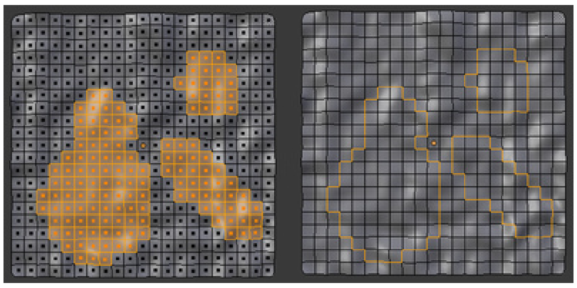
Mode: Edit mode → Edge select mode

Hotkey: CtrlE → Select Boundary Loop

Menu: Select » Select Boundary Loop or Mesh » Edges » Select Boundary Loop

Select Boundary Loop is the “logical inverse” of Select Loop Inner-Region, based on all regions currently selected, it selects only the edges at the border of these regions. It can operate in any select mode, but will always switch to Edge select mode when run.

All this is much more simple to illustrates with examples:


Select Boundary Loop does the opposite and forces into Edge Select Mode

Selecting Edges

Edges can be selected in much the same way as vertices and faces - by right-clicking them while Edge Select Mode is activated. Pressing ⇧ Shift while clicking will add/subtract to the existing selection.

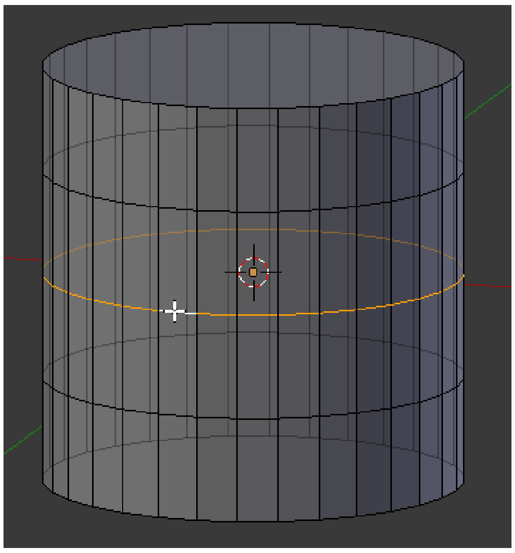
Edge Loops

Mode: Edit Mode (Mesh)

Hotkey: Alt RMB - or ⇧ ShiftAlt RMB for modifying existing selection

Menu: Select » Edge Loop

Edge loops can be selected by first selecting an edge (vertex or edge selection mode), and then going to Select » Edge Loop. The shortcut Alt RMB on an edge (either vertex or edge select mode) is a quicker and more powerful way of doing so. More powerful, because you can add/remove loops from an existing selection if you press ⇧ Shift too.
Note, that if you want to select a loop while being in vertex select mode, you still have to perform the shortcut on an edge - while you, for just selecting vertices, would rightclick on a vertex.



Alt on Linux
Alt is on some Linux distros caught by the windows manager. If you see the above shortcut not working, make sure that blender can properly recognize the usage of Alt.

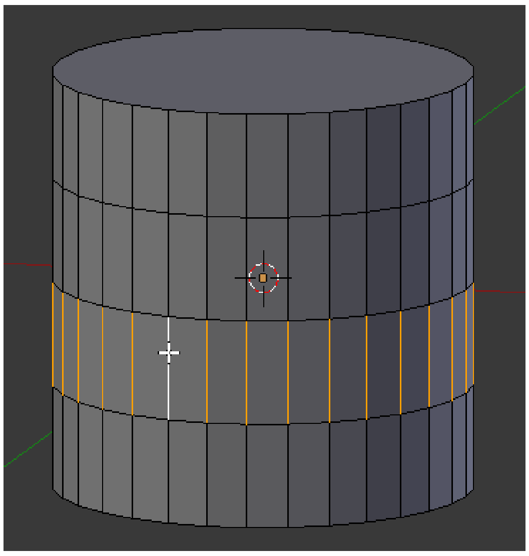
Edge Rings

Mode: Edit Mode (Mesh)

Hotkey: AltCtrl RMB - or ⇧ ShiftAltCtrl RMB for modifying existing selection

Menu: Select » Edge Ring

Edge Rings are selected similarly. Based on the selection of an edge go to Select » Edge Ring. Or use AltCtrl RMB on an edge.



Convert selection to whole faces
If the edge ring selection happened in Edge Select Mode, switching to Face Select Mode will erase the selection.

This is because none of those faces had all its (four) edges selected, just two of them.

Instead of selecting the missing edges manually or by using ⇧ ShiftAlt RMB twice, it is easier to first switch to Vertex Select Mode, which will kind of "flood" the selection. A subsequent switch to Face Select Mode will then properly select the faces.

Selecting Faces

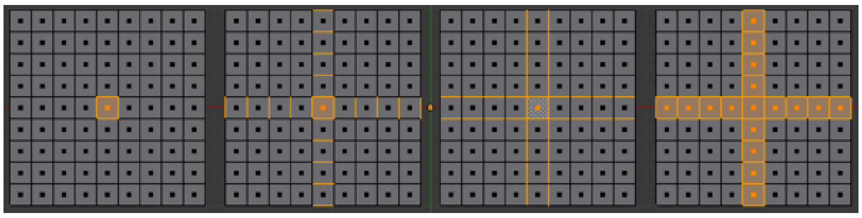
To select parts of a mesh face-wise, you have to switch to Face Select Mode. Do this by clicking the button shown above, or press Ctrl⇆ Tab to spawn a menu. The selection works as usual with RMB ; to add/remove to an existing selection, additionally press ⇧ Shift

[edit] Face Loops

Mode: Edit Mode (Mesh)

Hotkey: Alt RMB - or ⇧ ShiftAlt RMB for modifying existing selection

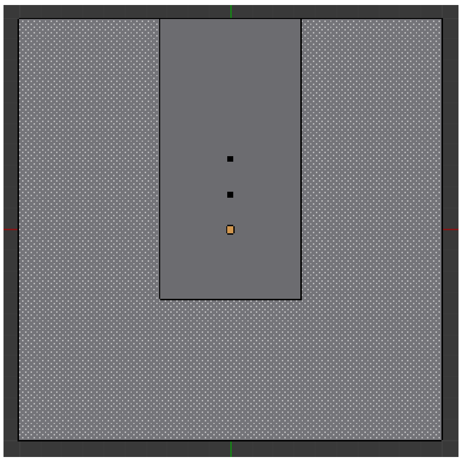
Face Loops are pretty much the same as Edge Rings. If you want to select a Face Loop, there is no menu entry that works based on a selected face. Using Select » Edge Ring would select a "cross" with the prior selected face as the middle. If you want to avoid switching to Edge Select Mode to select a Face Loop, use the Alt RMB shortcut.



Note: *Numbers correspond to images ordered from left to right*

1. Just the selected face.
2. Select the face, then Select » Edge Ring. See, how Blender selects edges, even if being in Face Select Mode. If these edges are desired and you want to work on them, switch to Edge Select Mode. Switching to Vertex Select Mode would flood the selection and leave you with the 4th image as result, that is, after having gone back to Face Select Mode after Vertex Select Mode.
3. Select the face, the Select » Edge Loop. As in the example above, Blender pretends to be in Edge Select Mode and takes the four edges of the selected face as base for the selection operation.
4. This selection was created by Alt RMB on the left edge of the center face, followed by twice ⇧ ShiftAlt RMB on the top edge of the center face. Two times, because the first click will remove the selected face loop (in this case, just the original selected face), while the second click will add the whole vertical running loop to the selection, creating the cross.

[edit] Ngons in Face Select Mode



As already known, faces are marked with a little square dot in the middle of the face. With ngons that can lead in certain cases to a confusing display. The example shows the center dot of the U-shaped ngon being inside of the oblong face inside the "U". It is not easy to say which dot belongs to which face (the orange dot in the image is the object center). Luckily, you don't need to care much - because to select a face, you don't have to click the center dot, but the face itself.

Face selection

To select a face:
Click the face, not the dot!

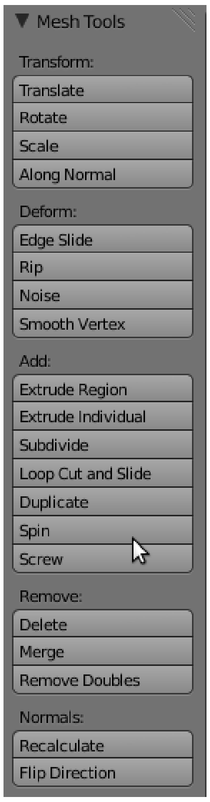
Mesh Editing

Blender provides a variety of tools for editing meshes. These are available through the Mesh Tools palette, the Mesh menu in the 3d view header, and context menus in the 3d view, as well as individual shortcut keys.

Note that all the “transform precision/snap” keys (Ctrl and/or ⇧ Shift) work also for all these advanced operations… However, most of them do not have axis locking possibilities, and some of them do not take into account pivot point and/or transform orientation either…

These transform tools are available in the Transform section of the Mesh menu in the menu bar. Note that some of these can also be used on other editable objects, like curves, surfaces, and lattices.

Types of Tools



Mesh Tools

The mesh tools are found in various places, and available through shortcuts as well.

Transform and Deform tools:

- Translate
- Rotate
- Scale
- Mirror
- Shrink/Flatten/Along Normal
- Push/Pull
- To Sphere
- Shear
- Warp
- Edge Slide
- Vertex Slide
- Noise
- Smooth Vertex
- Rotate Edge

Merge and Remove tools:

- Delete
- Dissolve
- Merge
- Auto-Merge
- Remove Doubles
- Tris to Quads
- Unsubdivide

Add and Divide tools:

- Make Edge/Face
- Fill
- Beauty Fill
- Solidify
- Quads to Tris
- Extrude Region
- Extrude Individual
- Subdivide
- Loop Cut/Slide
- Knife tool
- Vertex connect
- Duplicate
- Spin
- Screw
- Symmetrize
- Inset
- Bevel
- Wireframe

Separate tools:

- Rip
- Rip fill
- Split
- Separate
- Edge Split

Accessing Mesh Tools

Mesh Tools Palette

When you select a mesh and ⇆ Tab into edit mode, the Tool Shelf changes from Object Tools to Mesh Tools. These are only some of the mesh editing tools.

Menus

The Mesh is located in the Header bar. Some of the menus can be accessed with shortcuts:

CtrlF brings up the Face tool menu
CtrlE brings up the Edge tool menu
CtrlV brings up the Vertex tool menu

Basic Mesh Editing

In this section we explain how to do basic editing on a mesh.

Translation, Rotation, Scale

Mode: Edit mode

Panel: Mesh Tools (Editing context)

Hotkey: G/R/S

Menu: Mesh » Transform » Grab/Move, Rotate, Scale, …

Once you have a selection of one or more elements, you can grab/move (G), rotate (R) or scale (S) them, like many other things in Blender, as described in the Manipulation in 3D Space section.

To move, rotate and scale selected components, either use the Translate, Rotate, and Scale buttons, the transform manipulators, or the shortcuts:

G, R, and S respectively.

After moving a selection, the options in the Tool Shelf allow you to fine-tune your changes, limit the effect to certain axes, turn proportional editing on and off, etc.

Of course, when you move an element of a given type (e.g. an edge), you also modify the implicitly related elements of other kinds (e.g. vertices and faces).

You also have in Edit mode an extra option when using these basic manipulations: the proportional editing.

This page is being developed right now, follow this link to see the page while it's being constructed.

Page status (reviewing guidelines)

Void page
Proposed fixes: X

Object mode Adding object - menu, manual selection - is under last added object... or shortcut ⇧ ShiftA.

Edit mode: adding object -> addin mesh, e.g. final merging into one object.

center for adding: X 3D curson. If you need to center cursor imed. just press ⇧ ShiftC, or edit in the N tools menu cursor xoordinates x,y,z. Content proposal from Quark66 12:53, 11 June 2013 (CEST):

Deleting and Merging

These tools can be used to remove components.

Delete

Delete (X or Del)
: Deletes selected vertices, edges, or faces. This operation can also be limited to:

  Vertices
  : Delete all vertices in current selection, removing any faces or edges they are connected to.

  Edges
  : Deletes any edges in the current selection. Removes any faces that the edge shares with it.

  Faces
  : Removes any faces in current selection.

  Only Edges & Faces
  : Limits the operation to only selected edges and adjacent faces.

  Only faces
  : Removes faces, but edges within face selection are retained.

  Edge Collapse
  : Collapses edges into single vertices. This can be used to remove a loop of faces.

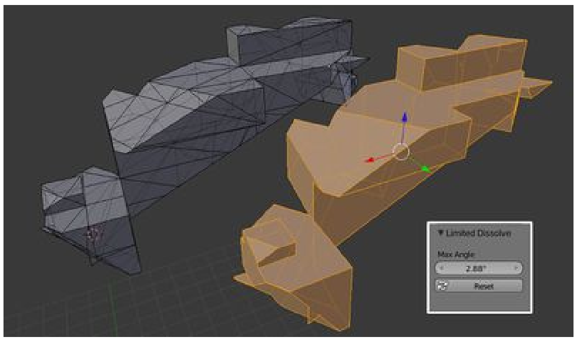
  Edge Loop
  : Deletes an edge loop. If the current selection is not an edge loop, this operation does nothing.

Dissolve

Dissolve operations are also accessed from the delete menu. Instead of removing the geometry, which may leave holes that you have to fill in again, dissolve will remove the geometry and fill in the surrounding geometry.

Dissolve
: Removes selected geometry, but keeps surface closed, effectively turning the selection into a single n-gon. Dissolve works slightly different based on if you have edges, faces or vertices selected. You can add detail where you need it, or quickly remove it where you don't.

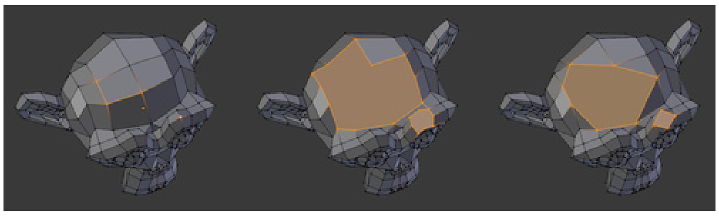
Limited Dissolve
: Limited Dissolve reduces detail on planar faces and linear edges with an adjustable angle threshold.



Max Angle: 2.88°

Example showing the how Limited Dissolve can be used.

Face Split - dissolve option.
: When dissolving vertices into surrounding faces, you can often end up with very large, uneven ngons.
  The face split option limits dissolve to only use the corners of the faces connected to the vertex.



Dissolve Face Split option. Left - the input, middle - regular dissolve, right - Face Split enabled

Convert Triangles to Quads

Tris to Quads AltJ This takes adjacent tris and removes the shared edge to create a quad. This tool can be performed on a selection of multiple triangles.

This same action can be done on a selection of just 2 tris, by selecting them and using the shortcut F, to create a face.

Unsubdivide

Mode: Edit mode

Hotkey: CtrlE » Unsubdivide

Menu: Mesh » Edges » Unsubdivide

Unsubdivide functions as the reverse of subdivide by attempting to remove edges that were the result of a subdivide operation. If additional editing has been done after the subdivide operation, unexpected results may occur.

Iterations
    How many subdivisions to remove.

Merging

Merging Vertices

Mode: Edit mode

Hotkey: AltM

Menu: Mesh » Vertices » Merge..., Specials » Merge or Vertex Specials » Merge

This tool allows you to merge all selected vertices into an unique one, deleting all others. You can choose the location of the surviving vertex in the menu this tool pops up before executing:

At First
    Only available in Vertex select mode, it will place the remaining vertex at the location of the first one selected.

At Last
    Only available in Vertex select mode, it will place the remaining vertex at the location of the last one selected (the active one).

At Center
    Available in all select modes, it will place the remaining vertex at the center of the selection.

At Cursor
    Available in all select modes, it will place the remaining vertex at the 3D Cursor.

Collapse
    This is a special option, as it might let “live” more than one vertex. In fact, you will have as many remaining vertices as you had “islands” of selection (i.e. groups of linked selected vertices). The remaining vertices will be positioned at the center of their respective “islands”. It is also available *via* the Mesh » Edges » Collapse menu option…

Merging vertices of course also deletes some edges and faces. But Blender will do everything it can to preserve edges and faces only partly involved in the reunion.

AutoMerge Editing

Mode: Edit mode

Menu: Mesh » AutoMerge Editing

The Mesh menu as a related toggle option: AutoMerge Editing. When enabled, as soon as a vertex moves closer to another one than the Limit setting (Mesh Tools panel, see below), they are automatically merged.

Remove Doubles

Mode: Edit mode

Panel: Editing context → Mesh Tools

Hotkey: W » 4 or CtrlV » Remove doubles

Menu: Mesh » Vertices » Remove Doubles, Specials » Remove Doubles or Vertex Specials » Remove Doubles

Remove Doubles is a useful tool to simplify a mesh by merging vertices that are closer than a specified distance to each other. An alternate way to simplify a mesh is to use the Decimate modifier.

Merge Distance
: Sets the distance threshold for merging vertices, in Blender units.

Unselected
: Allows vertices in a selection to be merged with unselected vertices. When disabled, selected vertices will only be merged with other selected ones.

Make Edge/Face

Mode: Edit mode

Hotkey: F

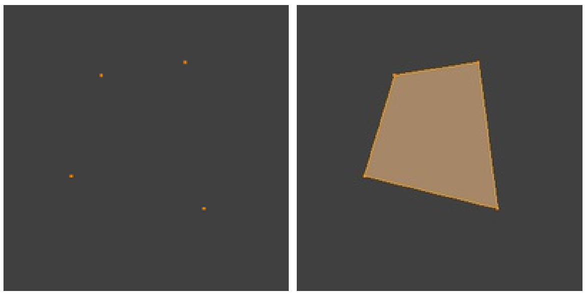
Menu: Mesh → Faces → Make Face/Edge

This is a context sensitive tool which creates geometry by filling in the selection. When only 2 vertices are selected it will create an edge, otherwise it will create faces.

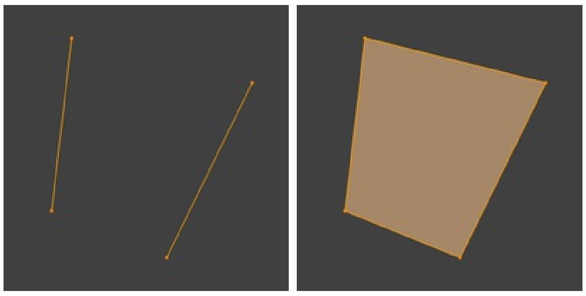
The typical use case is to select vertices and press F, however Blender also supports creating faces from different selections to help quickly build up geometry.

The following methods are used automatically depending on the context.

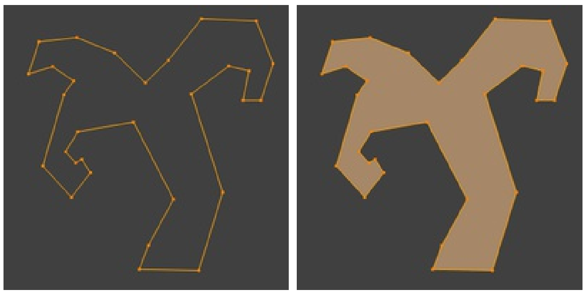
Isolated vertices.


Before After

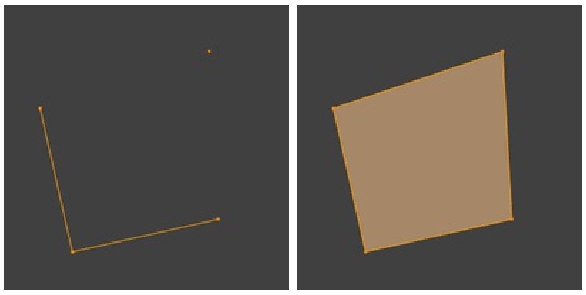
Isolated edges


Before After

N-gon from edges: *When there are many edges Blender will make an ngon, note that this doesn't support holes, to support holes you need to use the XXX Fill Faces tool.*

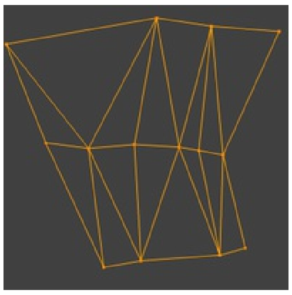
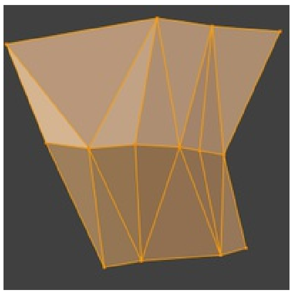

Before After

Mixed vertices/edges: *existing edges are used to make the face as well as an extra vertex.*

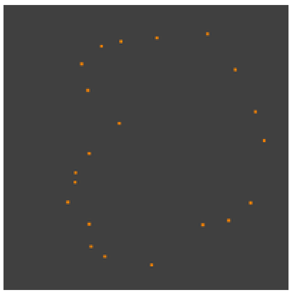
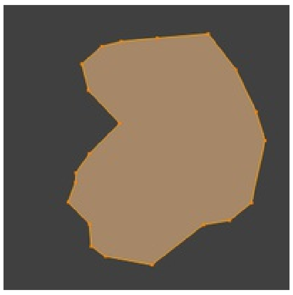


Before After

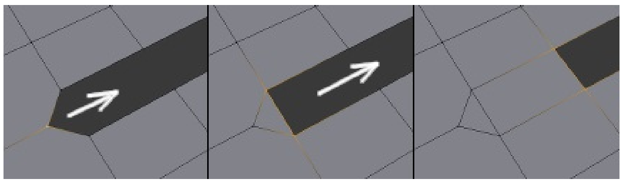
Edge-Net: *sometimes you may have many connected edges without interior faces.*



Before After

Point Cloud: *when there are many isolated vertices, Blender will calculate the edges for an n-gon.*



Before After

Single Vertex Selection: *with a single vertex selected on a boundary, the face will be created along the boundary, this saves manually selecting the other 2 vertices. Notice this tool can run multiple times to continue creating faces.*



Further Reading

For other ways to create faces see:

- Fill
- Grid Fill
- Bridge Edge Loops

Mirror Editing

X-Mirror

Mode: Edit mode

Panel: Mesh Options » X-mirror

The X-mirror option of the Mesh Options panel allows you edit both “sides” of your mesh in a single action. When you transform an element (vertex, edge or face), if there is its *exact X-mirrored counterpart* (in local space), it will be transformed accordingly, *through a symmetry along the local X axis*.

Topology Mirror

The Topology Mirror option is available in the 3D View Editor > Toolshelf Region > Mesh Options Panel whilst in Edit Mode

For Topology Mirror to work the X Mirror option must be enabled.

When using the X Mirror option to work on mirrored Mesh Geometry the vertices that are mirrored must be perfectly placed. If they are not exactly positioned in their mirror locations then X Mirror will not treat those vertices as mirrored. This can be annoying because often the out of position vertices are only very slightly out of position.

Topology Mirror tries to solve this problem by determining which vertices are mirrored vertices not only by using their positions but also by looking at how those vertices are related to others in the Mesh Geometry. It looks at the overall Mesh Geometry topology to determine if particular vertices will be treated as mirrored. The effect of this is that mirrored vertices can be non-symetrical and yet still be treated as mirrored when X Mirror and Topology Mirror are both active.

Note that Topology Mirror functionality will work more reliably on Mesh Geometry which is more detailed. If you use very simple Mesh Geometry such as a Cube or UV Sphere for example the Topology Mirror option will often not work.

For an example of how to use Topology Mirror open up a new Blender scene, then delete Blender's default cube and add a Monkey Object to the 3D Viewport.

Press the TAB Key to put the Monkey Object into Edit Mode.

With the X Mirror option disabled move one of the Monkey Object's vertices slightly.

Then Turn X Mirror option on again but leave Topology Mirror disabled

If you now move that vertice again X Mirror will not work and the mirrored vertices will not be altered.

If you then enable Topology Mirror and move the same vertices again, then X Mirror should still mirror the other vertice, even though they are not perfectly positioned.

Mirror Modifier

The conditions for X-mirror to work are quite strict, which can make it difficult to use. To have an exact mirrored version of a (half) mesh, its easier and simpler to use the Mirror modifier

Snap to Symmetry

Mode: Edit mode

Menu: Mesh » Snap to Symmetry

The Snap to Symmetry tool works on meshes which are mostly symmetrical but have vertices which have been moved enough that Blender does not detect them as mirrored through the x-mirror function.

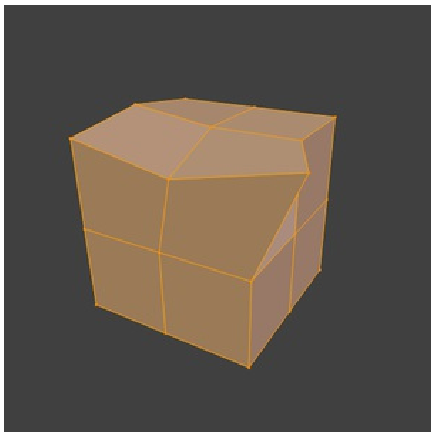
This can be caused by accident when editing without x-mirror enabled. Sometimes models imported from other applications are asymmetrical enough that mirror fails too.

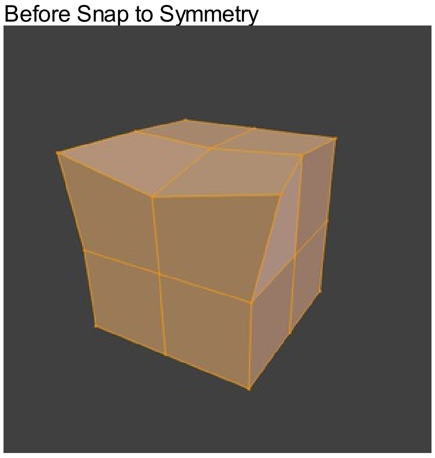
Direction
: Specify the axis and direction to snap. Can be any of the 3 axes, and either positive to negative, or negative to positive.

Threshold
: Specify the search radius to use when finding matching vertices.

Factor
: Support for blending mirrored locations from one side to the other (0.5 is an equal mix of both).

Center
: Snap vertices in the center axis to zero.


Before Snap to Symmetry


After Snap to Symmetry

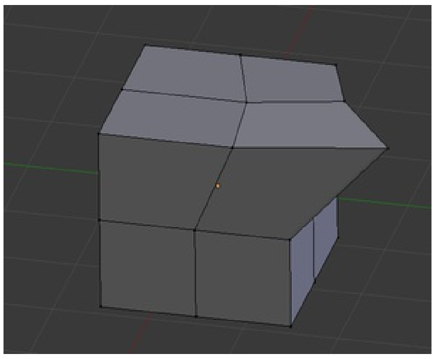
Symmetrize Mesh

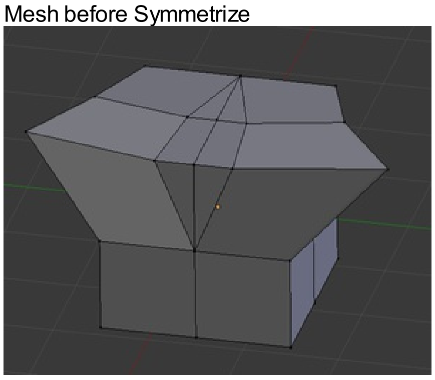
Mode: Edit mode

Menu: Mesh » Symmetrize

The Symmetrize tool is a quick way to make a mesh symmetrical. Symmetrize works by cutting the mesh at the pivot point of the object, and mirroring over the geometry in the specified axis, and merges the two halves together (if they are connected)

Direction
: Specify the axis and direction of the effect. Can be any of the 3 axes, and either positive to negative, or negative to positive.


Mesh before Symmetrize


Mesh after Symmetrize

Mirroring Geometry

See Mirror for information on mirroring, which allows you to flip geometry across an axis

Vertex Tools

This page covers many of the tools in the Mesh » Vertices menu. These are tools that work primarily on vertex selections, however, some also work with edge or face selections.

Merging

Merging Vertices

Mode: Edit mode

Hotkey: AltM

Menu: Mesh » Vertices » Merge..., Specials » Merge or Vertex Specials » Merge

This tool allows you to merge all selected vertices to an unique one, deleting all others. You can choose the location of the surviving vertex in the menu this tool pops up before executing:

At First
: Only available in Vertex select mode, it will place the remaining vertex at the location of the first one selected.

At Last
: Only available in Vertex select mode, it will place the remaining vertex at the location of the last one selected (the active one).

At Center
: Available in all select modes, it will place the remaining vertex at the center of the selection.

At Cursor
: Available in all select modes, it will place the remaining vertex at the 3D Cursor.

Collapse
: This is a special option, as it might let “live” more than one vertex. In fact, you will have as much remaining vertices as you had “islands” of selection (i.e. groups of linked selected vertices). The remaining vertices will be positioned at the center of their respective “islands”. It is also available *via* the Mesh » Edges » Collapse menu option…

Merging vertices of course also deletes some edges and faces. But Blender will do everything it can to preserve edges and faces only partly involved in the reunion.

AutoMerge Editing

Mode: Edit mode

Menu: Mesh » AutoMerge Editing

The Mesh menu as a related toggle option: AutoMerge Editing. When enabled, as soon as a vertex moves closer to another one than the Limit setting (Mesh Tools panel, see below), they are automatically merged.

Remove Doubles

Mode: Edit mode

Panel: Editing context → Mesh Tools

Hotkey: W » 4 or CtrlV » Remove doubles

Menu: Mesh » Vertices » Remove Doubles, Specials » Remove Doubles or Vertex Specials » Remove Doubles

Remove Doubles is a useful tool to simplify a mesh by merging vertices that are closer than a specified distance to each other. An alternate way to simplify a mesh is to use the Decimate modifier.

Merge Distance
: Sets the distance threshold for merging vertices, in Blender units.

Unselected
: Allows vertices in selection to be merged with unselected vertices. When disabled, selected vertices will only be merged with other selected ones.

Separating

Rip

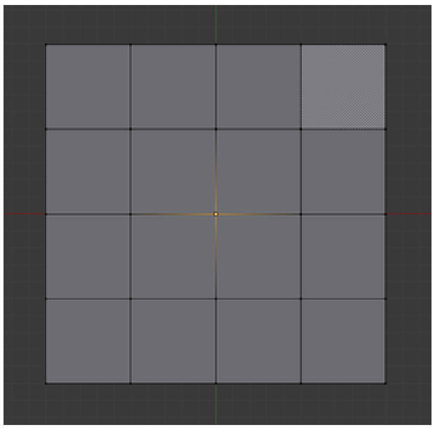
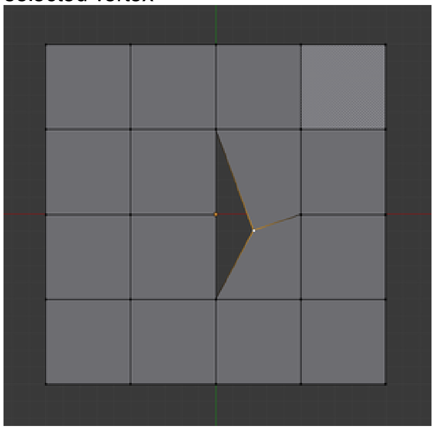
Mode: Edit mode

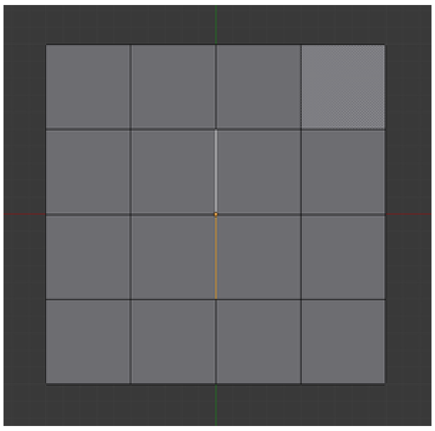
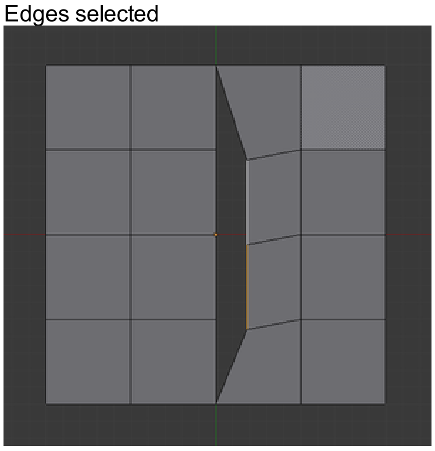
Hotkey: V

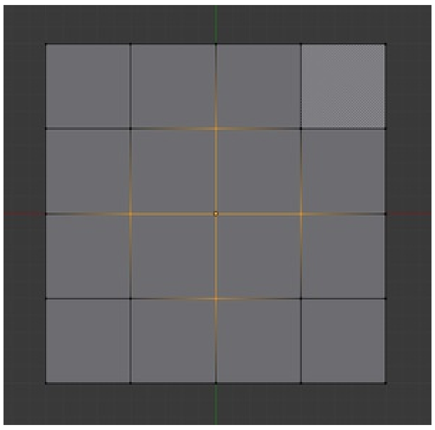
Menu: Mesh » Vertices » Rip

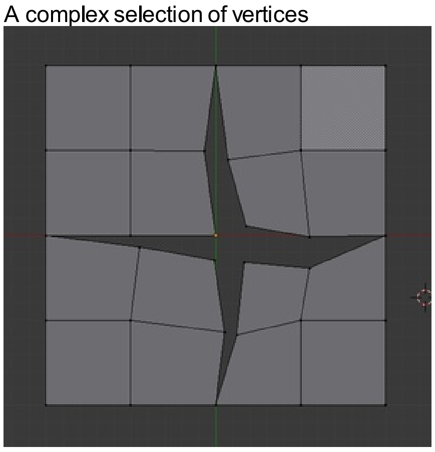
Rip creates a “hole” into a mesh by making a copy of selected vertices and edges, still linked to the neighbor non-selected vertices, so that the new edges are borders of the faces on one side, and the old ones, borders of the faces of the other side of the rip.

Examples


selected vertex

Hole created after using rip on vertex


Edges selected

Result of rip with edge selection


A complex selection of vertices


Result of rip operation

Limitations

Rip will only work when edges and/or vertices are selected. Using the tool when a face is selected (explicitly or implicitly), will return an error message “`Can’t perform ripping with faces selected this way`”. If your selection includes some edges or vertices that are not “between” two faces (manifold), it will also fail with message “`No proper selection or faces include`”.

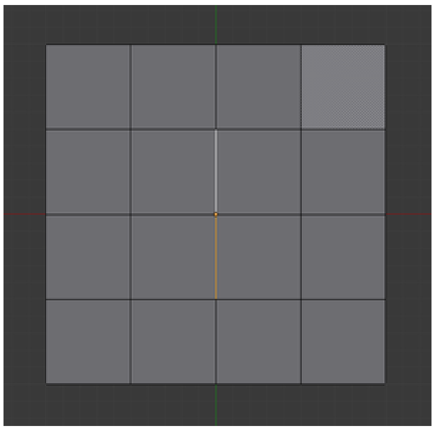
Rip Fill

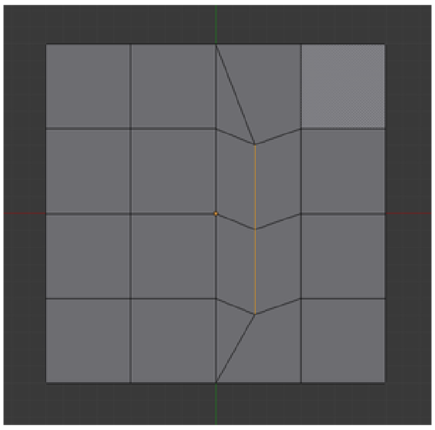
Mode: Edit mode

Hotkey: AltV

Menu: Mesh » Vertices » Rip Fill

Rip fill works the same as the Rip tool above, but instead of leaving a hole, it fills in the gap with geometry.


Edges selected


Result of rip fill

Split

Mode: Edit mode

Hotkey: Y

Menu: Mesh » Vertices » Split

A quite specific tool, it makes a sort of copy of the selection, removing the original data *if it is not used by any non-selected element*. This means that if you split an edge from a mesh, the original edge will still remain unless it is not linked to anything else. If you split a face, the original face itself will be deleted, but its edges and vertices remain unchanged. And so on.

Note that the “copy” is left exactly at the same position as the original, so you must move it (G) to see it clearly…

Separate

Mode: Edit mode

Hotkey: P

Menu: Mesh » Vertices » Separate

This will separate the selection in another mesh object, as described here.

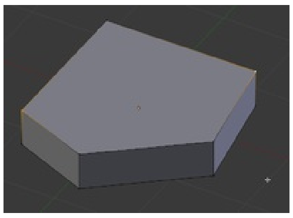
Vertex Connect

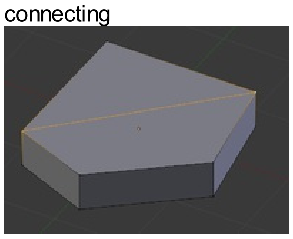
Mode: Edit mode

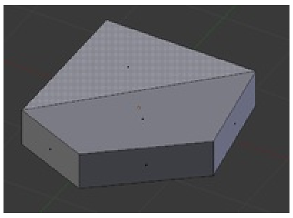
Hotkey: J

Menu: Mesh » Vertices » Vertex Connect or CtrlV » Vertex Connect

Vertex Connect takes two vertices that share a face, and creates an edge between the two, splitting the face into two new faces.


Selected vertices before connecting


After connecting vertices


Two faces created from
vertex connect operation

Vertex Slide

Mode: Edit mode

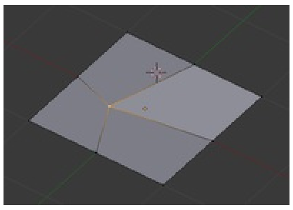
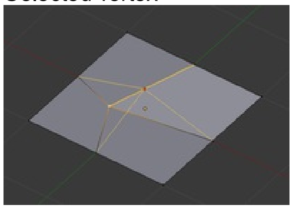
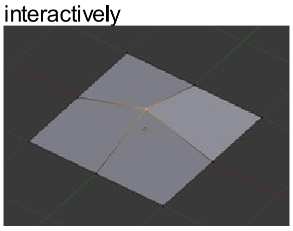
Panel: Editing context → Mesh Tools

Hotkey: ⇧ ShiftV » Vertex Slide

Menu: Mesh » Vertices » Vertex Slide or CtrlV » Vertex Slide

Vertex Slide will transform a vertex along one of its adjacent edges. Use ⇧ ShiftV to access the tool. Highlight the desired edge by moving the mouse, then confirm with LMB. Drag the cursor to specify the position along the line formed by the edge, then LMB again to move the vertex. There are three options available by holding the following keys:

Snap to Midpoint ⇧ Shift
Snap to Endpoint Alt
Snap and Merge to Endpoint Control


Selected vertex

Positioning vertex
interactively

Repositioned vertex

Smooth

Mode: Edit mode

Panel: Editing context → Mesh Tools

Hotkey: CtrlV » Smooth vertex

Menu: Mesh » Vertices » Smooth, Specials » Smooth or Vertex Specials » Smooth

This will apply once the Smooth Tool.

Make Vertex Parent

Mode: Edit mode

Hotkey: CtrlP

Menu: Mesh » Vertices » Make Vertex Parent

This will parent the other selected object(s) to the vertices/edges/faces selected, as described here.

Add Hook

Mode: Edit mode

Hotkey: CtrlH

Menu: Mesh » Vertices » Add Hook

Adds a Hook Modifier (using either a new empty, or the current selected object) linked to the selection. Note that even if it appears in the history menu, this action cannot be undone in Edit mode – probably because it involves other objects…

Blend From Shape, Propagate Shapes

Mode: Edit mode

Hotkey: W » AltBlend from shape or CtrlV » Blend From Shape, and W » AltShape propagate or CtrlV » Shape Propagate

Menu: (Vertex) Specials » Blend From Shape and Vertex Specials » Shape Propagate

These are options regarding shape keys.

Make Edge/Face

Mode: Edit mode

Hotkey: F

Menu: Mesh » Edges » Make Edge/Face

It will create an edge or some faces, depending on your selection. We have already discussed this tool in the editing basics page.

Set Edge Attributes

Edges can have several different attributes that influence how varying tools affect the mesh.

Mark Seam and Clear Seam

Mode: Edit mode (Vertex or Edge select modes)

Hotkey: CtrlE6 NumPad and CtrlE7 NumPad

Menu: Mesh » Edges » Mark Seam/Clear Seam (or the same options in Edge Specials menu)

Seams are a way to create separations, "islands", in UV maps. See the UVTexturing section for more details. These commands set or unset this flag for selected edges.

Mark Sharp and Clear Sharp

Mode: Edit mode (Vertex or Edge select modes)

Hotkey: CtrlE8 NumPad and CtrlE9 NumPad

Menu: Mesh » Edges » Mark Seam/Clear Seam (or the same options in Edge Specials menu)

The Sharp flag is used by the EdgeSplit modifier, which is part of the various smoothing techniques. Like seams, it is a property of edges, and these commands set or unset it for selected ones.

Adjust Bevel Weight

Mode: Edit mode (Vertex or Edge select modes)

Hotkey: Ctrl⇧ ShiftE

Menu: Mesh » Edges » Adjust Bevel Weight

This edge property (a value between **0.0** and **1.0**) is used by the Bevel modifier to control the bevel intensity of the edges. This command enters an interactive mode (a bit like transform tools), where by moving the mouse (or typing a value with the keyboard) you can set the (average) bevel weight of selected edges.

Crease SubSurf

Mode: Edit mode (Vertex or Edge select modes)

Hotkey: ⇧ ShiftE

Menu: Mesh » Edges » Crease SubSurf

This edge property (a value between **0.0** and **1.0**) is used by the Subsurf modifier to control the sharpness of the edges in the subdivided mesh. This command enters an interactive mode (a bit like transform tools), where by moving the mouse (or typing a value with the keyboard) you can set the (average) crease value of selected edges. To clear the crease edge property, enter a value of **-1**.

Edge Slide

Mode: Edit mode (Vertex or Edge select modes)

Hotkey: CtrlE » 6 NumPad

Menu: Mesh » Edges » Slide Edge (or the same option in Edge Specials menu)

Slides one or more edges across adjacent faces with a few restrictions involving the selection of edges (i.e. the selection must make sense, see below.)

EvenE
: Forces the edge loop to match the shape of the adjacent edge loop. You can flip to the opposite vertex using F. Use Alt Wheel to change the control edge.

Flip F
: When Even mode is active, this flips between the two adjacent edge loops the active edge loop will match

LMB confirms the tool, and RMB or Esc cancels.

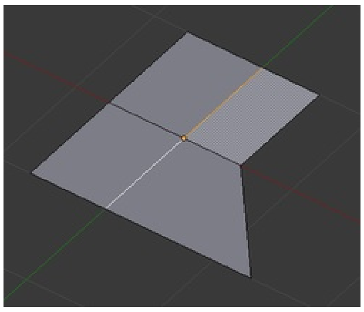
This tool has a factor, which is displayed in the 3D View footer and in the Tool Shelf (after confirmation). A numerical value between `-1` and `1` can be entered for precision.

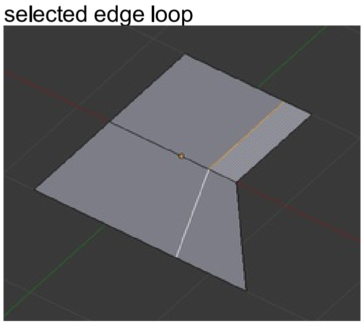
In *Proportional* mode, Wheel, or ← and → changes the selected edge for calculating a proportion. Unlike *Percentage* mode, *Proportional*

Holding Ctrl or ⇧ Shift control the precision of the sliding. Ctrl snaps movement to 10% steps per move and ⇧ Shift snaps movement to 1% steps. The default is 5% steps per move.

Usage

By default, the position of vertices on the edge loop move as a percentage of the distance between their original position and the adjacent edge loop, regardless of the edges' lengths.


selected edge loop

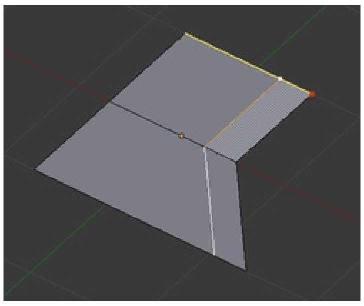

Repositioned edge loop

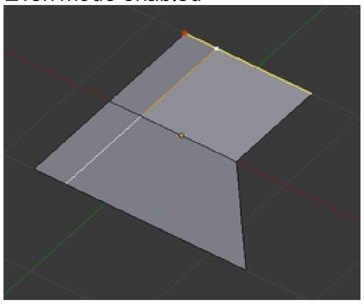
Even mode

Even mode keeps the shape of the selected edge loop the same as one of the edge loops adjacent to it, rather than sliding a percentage along each perpendicular edge.

In *Even* mode, the tool shows the position along the length of the currently selected edge which is marked in yellow, from the vertex that is an enlarged red marker. Movement of the sliding edge loop is restricted to this length. As you move the mouse, the length indicator in the header changes showing where along the length of the edge you are.

To change the control edge that determines the position of the edge loop, use the Alt Wheel to scroll to a different edge.


Even mode enabled


Even mode with flip enabled

Moving the mouse moves the selected edge loop towards or away from the start vertex, but the loop line will only move as far as the length of the currently selected edge, conforming to the shape of one of the bounding edge loops.

Limitations & Workarounds

There are restrictions on the type of edge selections that can be operated upon. Invalid selections are:

Loop crosses itself
: This means that the tool could not find any suitable faces that were adjacent to the selected edge(s). (*Loop crosses*) is an example that shows this by selecting two edges that share the same face. A face cannot be adjacent to itself.

Multiple edge loops
: The selected edges are not in the same edge loop, which means they don't have a common edge. You can minimize this error by always selecting edges end to end or in a "Chain". If you select multiple edges just make sure they are connected. This will decrease the possibility of getting looping errors.

Border Edge
: When a single edge was selected in a single sided object. An edge loop can not be found because there is only one face. Remember, edge loops are loops that span two or more faces.

A general rule of thumb is that if multiple edges are selected they should be connected end to end such that they form a continuous chain. This is *literally* a general rule because you can still select edges in a chain that are invalid because some of the edges in the chain are in different edge loops.

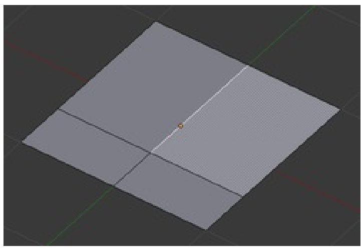
Rotate Edge

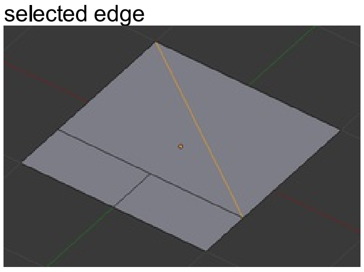
Mode: Edit mode (Vertex or Edge select modes)

Hotkey: CtrlE » Rotate Edge CW and CtrlE » Rotate Edge CCW

Menu: Mesh » Edges » Rotate Edge CW / Rotate Edge CCW

Rotating an edge clockwise or counter-clockwise spins an edge between two faces around their vertices. This is very useful for restructuring a mesh's topology. The tool can operate on one explicitly selected edge, or on two selected vertices or two selected faces that implicitly share an edge between them.


selected edge


Edge, rotated CW

Using Face Selection

To rotate an edge based on faces you must select two faces, (*Adjacent selected faces*), otherwise Blender notifies you with an error message, "`ERROR: Could not find any select edges that can be rotated`". Using either Rotate Edge CW or Rotate Edge CCW will produce exactly the same results as if you had selected the common edge shown in (*Selected edge rotated CW and CCW.*).

Delete Edge Loop

Mode: Edit mode (Vertex or Edge select modes)

Hotkey: X/Del » G

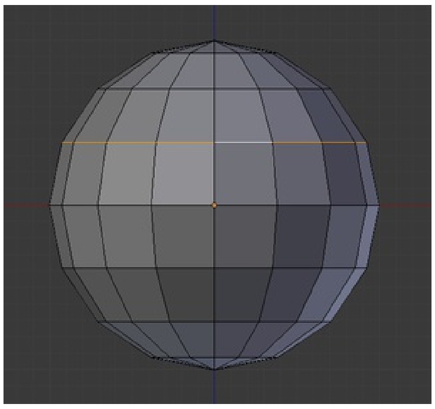
Menu: Mesh » Delete » Edge Loop

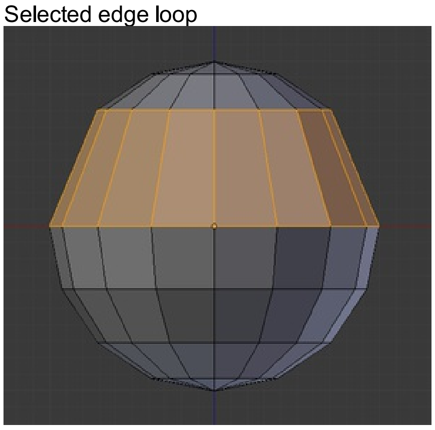
Delete Edge Loop allows you to delete a selected edge loop if it is between two other edge loops. This will create one face-loop where two previously existed.

Note
The Edge Loop option is very different to the Edges option, even if you use it on edges that look like an edge loop. Deleting an edge loop merges the surrounding faces together to preserve the surface of the mesh. By deleting a chain of edges, the edges are removed, deleting the surrounding faces as well. This will leave holes in the mesh where the faces once were.

Example

The selected edge loop on the UV Sphere has been deleted and the faces have been merged with the surrounding edges. If the edges had been deleted by choosing Edges from the (*Erase Menu*) there would be an empty band of deleted faces all the way around the sphere instead.


Selected edge loop


Edge loop deleted

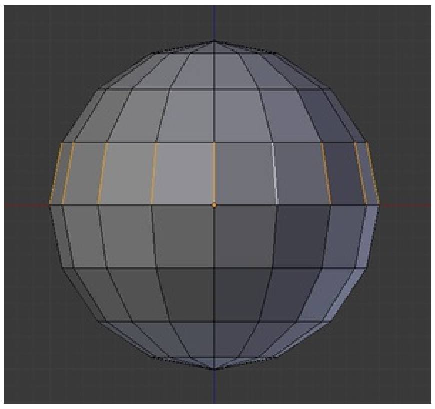
Collapse

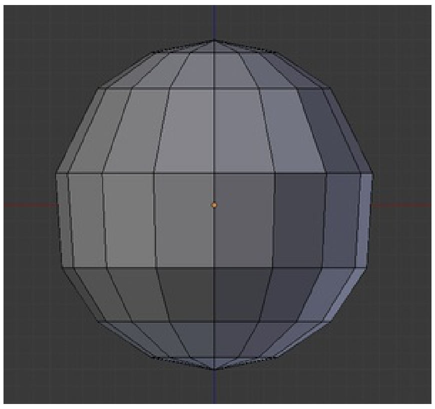
Mode: Edit mode

Hotkey: AltM » 3 NumPad

Menu: Mesh » Delete » Edge Collapse

This takes a selection of edges and for each edge, merges its two vertices together. This is useful for taking a ring of edges and collapsing it, removing the face loop it ran through.


Selected edge ring


Edge ring collapsed

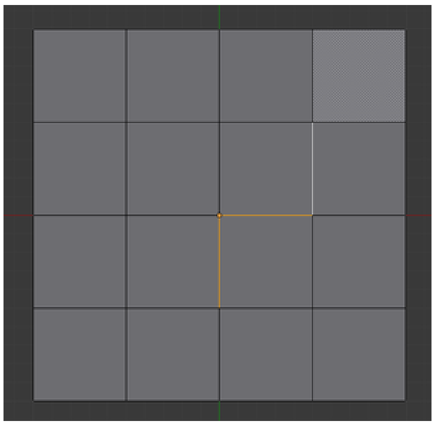
Edge Split

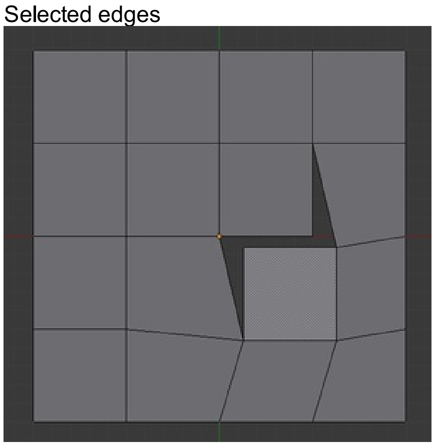
Mode: Edit mode

Hotkey: CtrlE » Edge Split

Menu: Mesh » Edges » Edge Split

Edge split is similar to the rip tool. When two or more touching interior edges, or a border edge is selected when using Edge split, a hole will be created, and the selected edges are duplicated to form the border of the hole


Selected edges

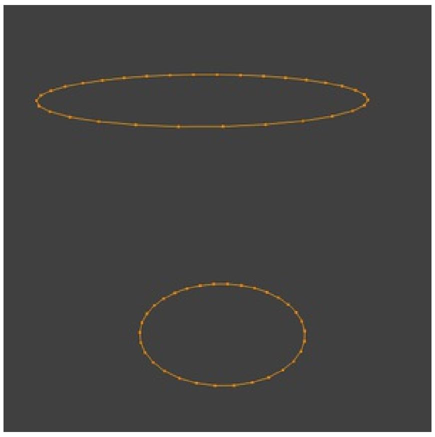

Adjacent face moved to reveal hole left by split

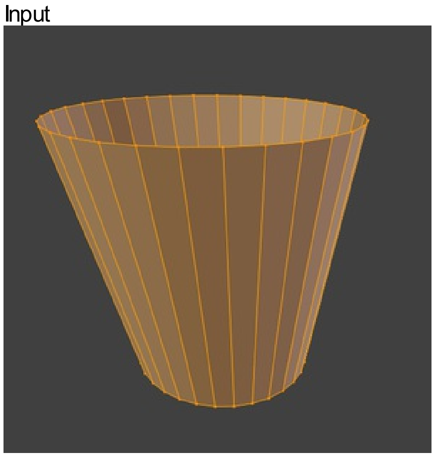
Bridge Edge Loops

Mode: Edit mode

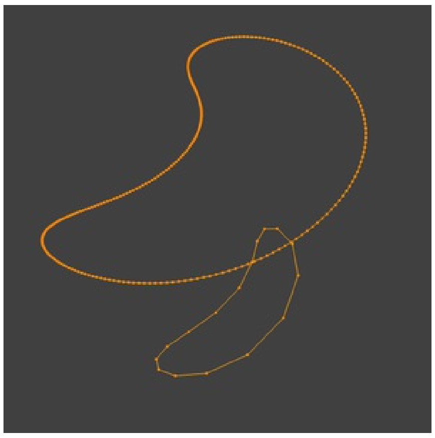
Menu: Mesh » Edges » Bridge Edge Loops

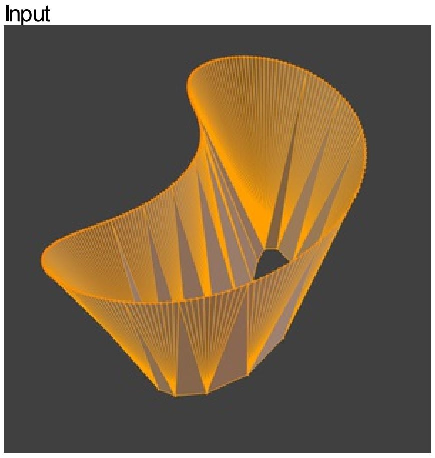
Bridge Edge Loops connects multiple edge loops with faces.
Simple example showing 2 closed edge loops.


Input

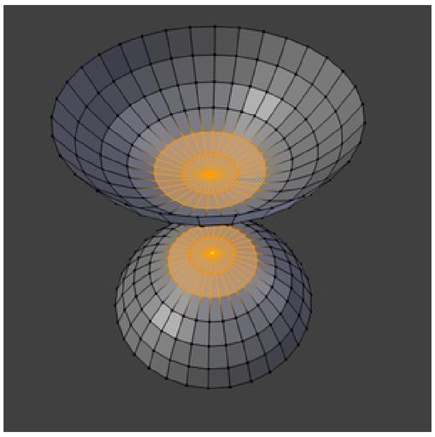

Bridge result

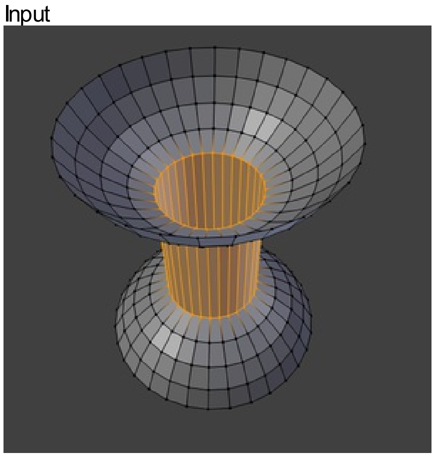
Example of bridge tool between edge loops with different numbers of vertices.


Input

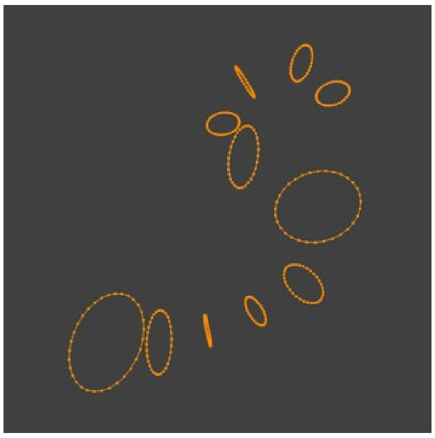

Bridge result

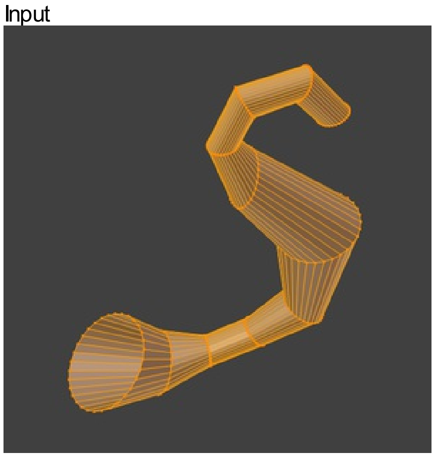
Example using the bridge tool to punch holes in face selections and connect them.


Input

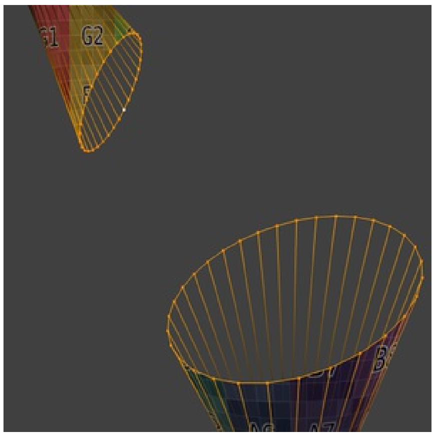

Bridge result

Example showing how bridge tool can detect multiple loops and loft them in one step.

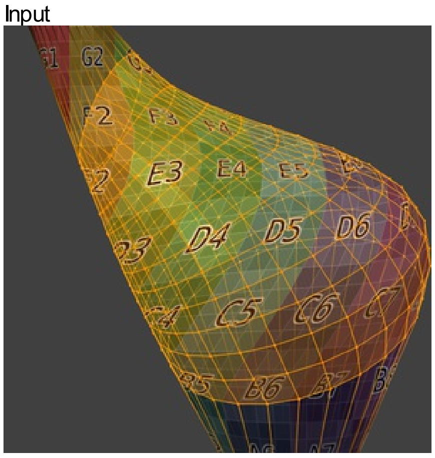

Input


Bridge result

Example of the subdivision option and surface blending with UV's.



Input



Bridge result

Face Tools

These are tools that manipulate faces.

Creating Faces

Make Edge/Face

Mode: Edit mode

Hotkey: F

Menu: Mesh » Faces » Make Edge/Face

This will create an edge or some faces, depending on your selection. It is detailed in the Basic Editing page.

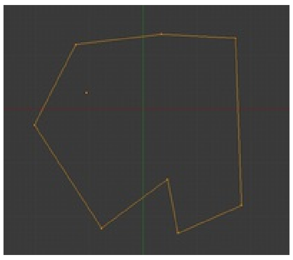
Fill

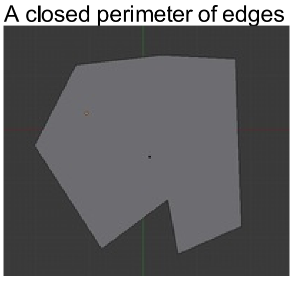
Mode: Edit mode

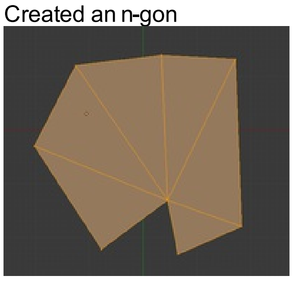
Hotkey: AltF

Menu: Mesh » Faces » Fill/Beautify Fill

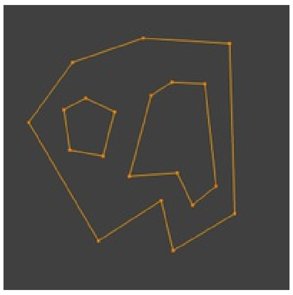
The Fill option will create *triangular* faces from any group of selected edges or vertices, *as long as they form one or more complete perimeters*.

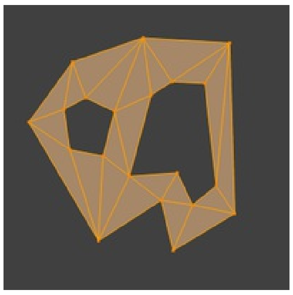

A closed perimeter of edges


Filled using shortcut F.
Created an n-gon


Filled using fillAltF

note, unlike creating n-gons, fill supports holes.


A closed perimeter of edges
with holes


Filled using fillAltF

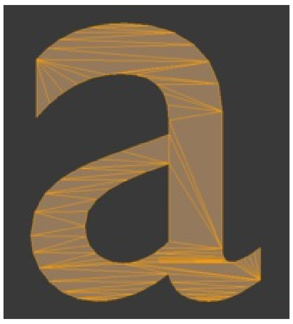
Beauty Fill

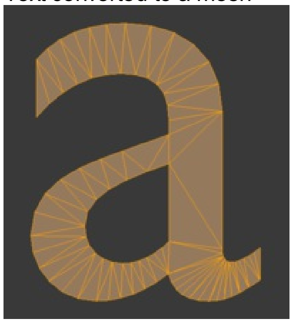
Mode: Edit mode

Hotkey: Alt⇧ ShiftF

Menu: Mesh » Faces » Fill/Beautify Fill

Beautify Fill works only on selected existing faces. It rearrange selected triangles to obtain more “balanced” ones (i.e. less long thin triangles).


Text converted to a mesh

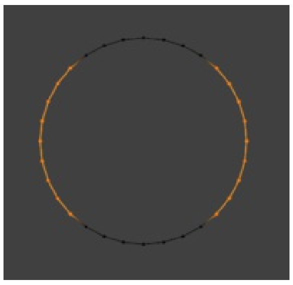

Result of Beauty Fill,Alt⇧ ShiftF

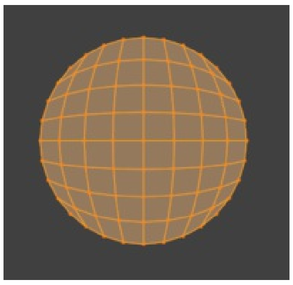
Grid Fill

Mode: Edit mode

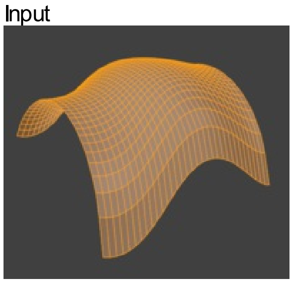
Menu: Mesh » Faces » Fill/Grid Fill

Grid Fill uses a pair of connected edge-loops to fill in a grid that follows the surrounding geometry.


Input


Grid fill result

Input


Grid fill result

Convert Quads to Triangles

Mode: Edit mode

Hotkey: CtrlT

Menu: Mesh » Faces » Convert Quads to Triangles or Face Specials » Triangulate

As its name intimates, this tool converts each selected quadrangle into two triangles. Remember that quads are just a set of two triangles.

Convert Triangles to Quads

Mode: Edit mode

Panel: Mesh Tools (Editing context)

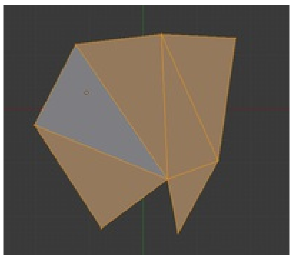
Hotkey: AltJ

Menu: Mesh » Faces » Convert Triangles to Quads

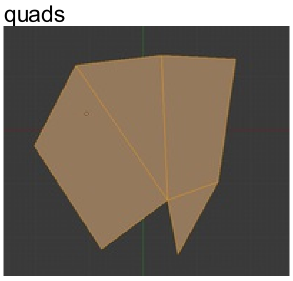
This tool converts the selected triangles into quads by taking adjacent tris and removes the shared edge to create a quad, based on a threshold. This tool can be performed on a selection of multiple triangles.

This same action can be done on a selection of 2 tris, by selecting them and using the shortcut F, to create a face, or by selecting the shared edge and dissolving it with the shortcut X » Dissolve.

To create a quad, this tool needs at least two adjacent triangles. If you have an even number of selected triangles, it is also possible not to obtain only quads. In fact, this tool tries to create “squarishest” quads as possible from the given triangles, which means some triangles could remain.


Before converting tris to

quads


After converting tris to quads, with a max angle of 30

All the menu entries and hotkey use the settings defined in the Mesh Tools panel:

Max Angle
: This values (between **0** and **180**) controls the threshold for this tool to work on adjacent triangles. With a threshold of **0.0**, it will only join adjacent triangles that form a perfect rectangle (i.e. right-angled triangles sharing their hypotenuses). Larger values are required for triangles with a shared edge that is small, relative to the size of the other edges of the triangles.

Compare UVs
: When enabled, it will prevent union of triangles that are not also adjacent in the active UV map. Note that this seems to be the only option working…

Compare Vcol
: When enabled, it will prevent union of triangles that have no matching vertex color. I'm not sure how this option works – or even if it really works…

Compare Sharp
: When enabled, it will prevent union of triangles that share a “sharp” edge. I'm not sure either if this option works, and what is the “sharp” criteria – neither the Sharp flag nor the angle between triangles seem to have an influence here…

Compare Materials
: When enabled, it will prevent union of triangles that do not use the same material index. This option does not seem to work neither…

Solidify

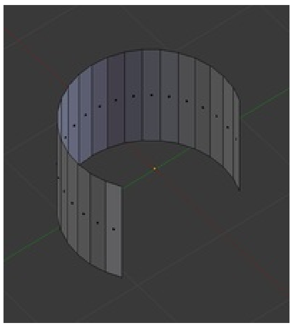
Mode: Edit mode

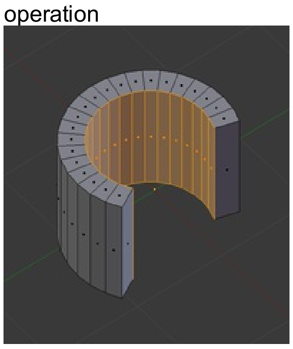
Hotkey: CtrlF » Solidify

Menu: Mesh » Faces » Solidify

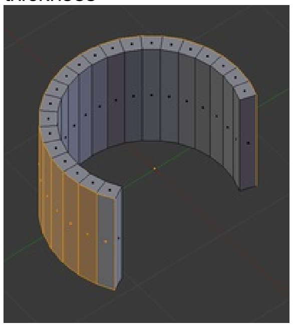
This takes a selection of faces and solidifies them by extruding them uniformly to give volume to a non-manifold surface. This is also available as a Modifier. After using the tool, you can set the offset distance in the Tool Palette.

Thickness
: Amount to offset the newly created surface. Positive values offset the surface inward relative to the normals. Negative values offset outward.


Mesh before solidify operation



Solidify with a positive thickness


Solidify with a negative thickness

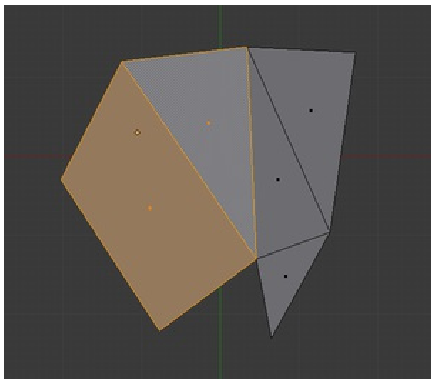
Rotate Edges

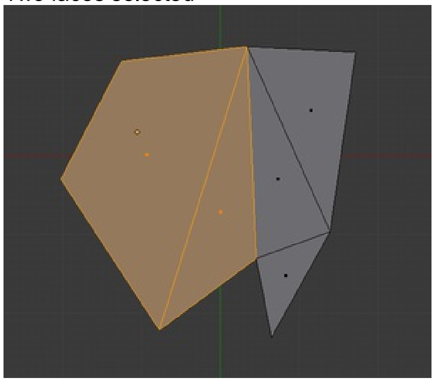
Mode: Edit mode

Menu: Mesh » Faces » Rotate Edge CW

This command functions the same edge rotation in edge mode.

It works on the shared edge between two faces and rotates that edge if the edge was selected.


Two faces selected


After rotating edge

See Rotate Edge CW / Rotate Edge CCW for more information.

Normals

As normals are mainly a face “sub-product”, we describe their few options here also.

See Smoothing for additional information on working with face normals.

Flip Direction

Mode: Edit mode

Hotkey: W » Flip Normals}

Menu: Mesh » Normals » Flip or Specials » Flip Normals

Well, it will just reverse the normals direction of all selected faces. Note that this allows you to precisely control the direction (**not the**

orientation, which is always perpendicular to the face) of your normals, as only selected ones are flipped.

Recalculate Normals

Mode: Edit mode

Hotkey: CtrlN and ctrl

Menu: Mesh » Normals » Recalculate Outside and Mesh » Normals » RecalculateInside

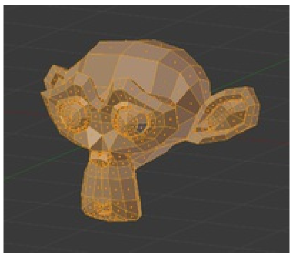
These commands will recalculate the normals of selected faces so that they point outside (respectively inside) the volume that the face belongs to. This volume do not need to be closed. In fact, this means that the face of interest must be adjacent with at least one non-coplanar other face. For example, with a Grid primitive, neither Recalculate Outside nor Recalculate Inside will never modify its normals…

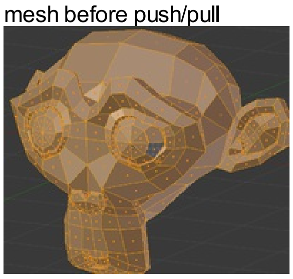
Deforming Geometry

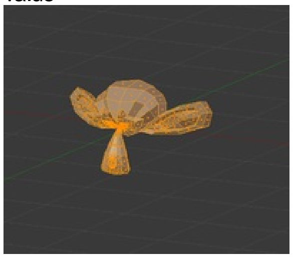
Push/Pull

Similar to shrink/flatten, this transformation consists of translating all selected elements along the line joining their original position to the Average position of the points. All translations are of same value, and are controlled by the mouse. It results in something that looks a bit like the scale effect, but much more deforming.

Note that unlike the preceding ones, you can lock this transformation on axis – even though this has no real interest (except perhaps with a “plane locking”…).


mesh before push/pull


Pulled out using a negative value


Pushed in using a positive value

Warp

Mode: Edit mode

Hotkey: ⇧ ShiftW

Menu: Mesh/Curve/Surface » Transform » Warp

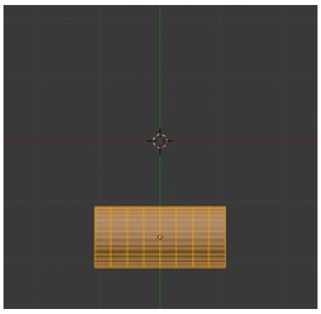
The Warp transformation is useful in very specific cases. It works by warping the selected elements around the 3D cursor (always the 3D cursor; it does not take into account the pivot point setting…). It is also view-dependent. The points that line up vertically with the cursor will remain in place. Each point's distance to the cursor's **horizontal position** corresponds to the radius it will be from the cursor after the tool has been activated. The tool will then wrap the point around the cursor. A value of 360 will wrap the mesh into a complete circle, so that the points furthest to the left and the right will line up with each other and the cursor position.

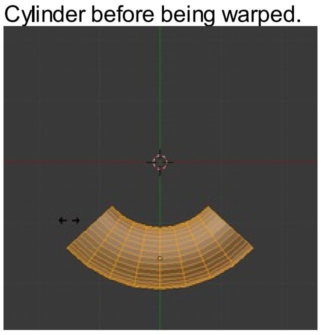
To use this tool, set the cursor in the view where the center should be. Activate the tool, then move the cursor or enter the value of the angle the mesh should be warped to.

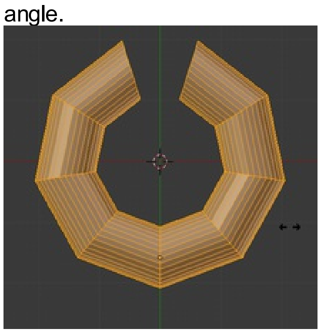
Example

A cylinder is warped into a semicircular shape

- Switch to top view and move the mesh away from the 3D cursor. This distance defines the radius of the warp.
- Place the mesh in Edit mode (⭾ Tab) and press A to select all vertices. Press ⇧ ShiftW to activate the warp transform tool. Move the mouse left or right to interactively define the amount of warp.


Cylinder before being warped.


Cylinder warped, using a small angle.


Warp using larger angle.

Shear

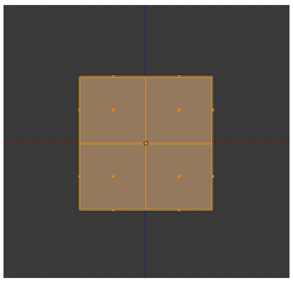
Mode: Edit mode

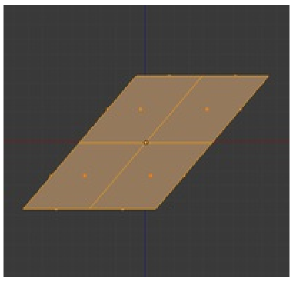
Hotkey: CtrlAlt⇧ ShiftS

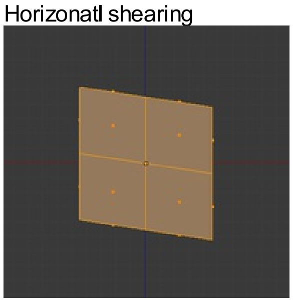
Menu: Object/Mesh/Curve/Surface » Transform » Shear

The Shear transformation applies a shearing on your selection of elements (in Edit mode, vertices/edges/control points/…). Like the other transform tools, it uses the view space, and is centered on the pivot point: the shear occurs along the view's x-axis passing through the pivot point. Everything that is "above" this axis (i.e. has a positive y-axis position) will move (shear) in the same direction as your mouse pointer (but always parallel to the x-axis). And everything that is "below" this x-axis will move in the opposite direction. The further away from the x-axis an element is, the more it moves.

When the tool becomes active, move the mouse left to right to interactively control the shearing. To make the effect work on the vertical axis instead of the horizontal one, click the MMB and then move the mouse up or down. Alternatively enter a numerical value from 0 to infinity. To finish with the tool, press the LMB.


before shearing


Horizonatl shearing


Vertical shearing

To Sphere

Mode: Edit modes

Panel: Mesh Tools (Editing context)

Hotkey: ⇧ ShiftAltS

Menu: Mesh/Curve/Surface » Transform » To Sphere

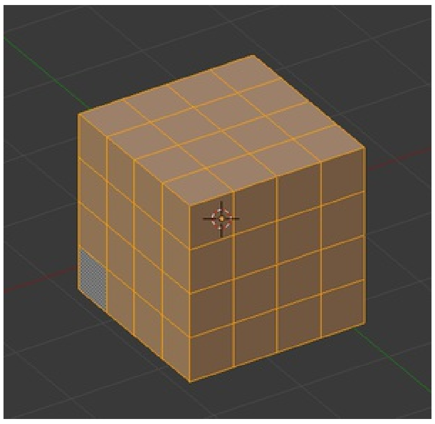
This command “spherifies” the selected mesh elements. It does this by finding the average position of the elements, and moves them toward the average distance they are from this point. Using a value of 1 puts all of the vertices an equal distance from this point, creating a spherical shape.

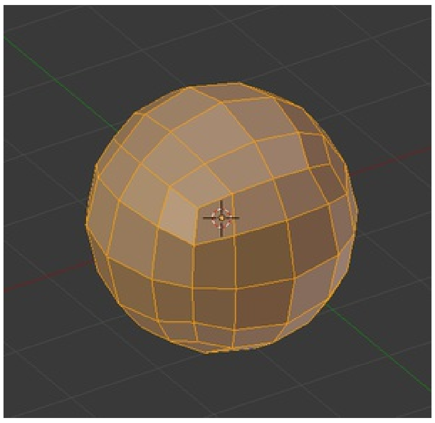
When the tool becomes active, drag the mouse left or right to interactively control the effect, or type in a value from 0 to 1 to manually control it.

Example

First, start with a Cube.

- Press ⇆ Tab to switch into Edit mode.
- Make sure all the vertices of the cube are selected by pressing A twice. Then, go to the Editing context by pressing F9. You should be able to see the Mesh Tools panel now.
- Subdivide the cube by pressing the Subdivide button in the Mesh Tools panel, or with W » Subdivide. You can do this as many times as you want; the more you subdivide, the smoother your sphere will be.
- Now, press ⇧ ShiftAltS and move your mouse left or right to interactively control the proportion of “spherification” (or directly type a value, like “1.000” to achieve the same effect as below) – preferably using the Median Point pivot point!
- Alternatively, you can use the To Sphere button (in the Mesh Tools panel). Select “100” to make your sphere. Note that you *should not move the 3D cursor* – or you won’t get a sphere, but a piece of sphere…


Subdivided cube, before


Subdivided cube, after warp

Mirror

Mode: Edit mode

Hotkey: CtrlM

Menu: Mesh » Mirror » `Desired Axis`

The mirror tool mirrors a selection across a selected axis.

The mirror tool in Edit mode is similar to Mirroring in Object mode. It is exactly equivalent to scaling by -1 vertices, edges or faces around one chosen pivot point and in the direction of one chosen axis, only it is faster/handier.

After this tool becomes active, select an axis to mirror the selection on entering x,y, or z.

You can also interactively mirror the geometry by holding the MMB and dragging in the direction of the desired mirror direction.

Axis of symmetry

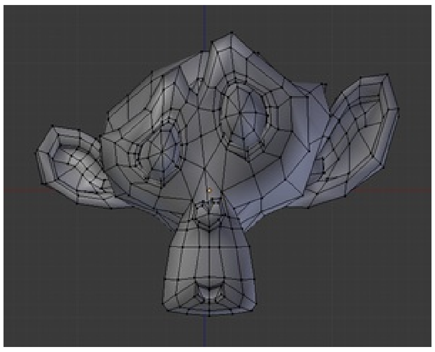
For each transformation orientation, you can choose one of its axes along which the mirroring will occur.

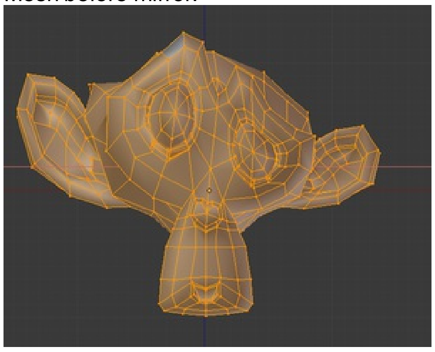
As you can see, the possibilities are infinite and the freedom complete: you can position the pivot point at any location around which we want the mirroring to occur, choose one transformation orientation and then one axis on it.

Pivot point

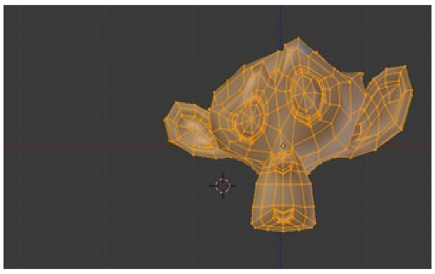
Pivot points must be set first. Pivot points will become the center of symmetry. If the widget is turned on it will always show where the pivot point is.

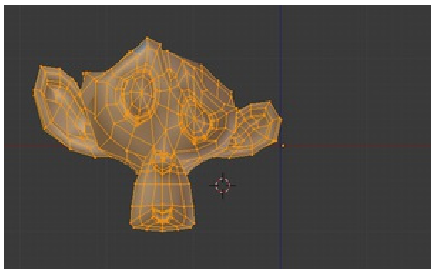
Note: The pivot point defaults to the **median point** in Edit mode. This is a special case of Edit mode as explained in the pivot point page.


Mesh before mirror.


Mesh after mirrored along X axis

On (*Mirror around the 3D Cursor…*) the pivot point is the 3D Cursor, the transformation orientation is Local, a.k.a. the Object space, and the axis of transformation is X.


Mesh before mirror.


Mesh after mirrored along X axis using the 3d cursor as a pivot point

Transformation orientation

Transformation Orientations are found on the 3D area header, next to the Widget buttons. They decide which coordinate system will rule the mirroring.

Shrink/Fatten Along Normals

Mode: Edit mode

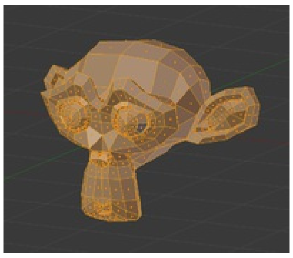
Panel: Mesh Tools (Editing context)

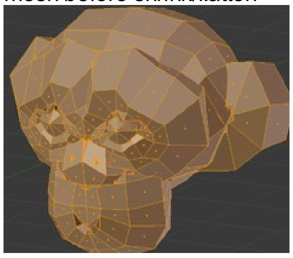
Hotkey: AltS

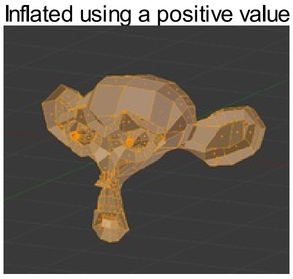
Menu: Mesh » Transform » Shrink/Fatten Along Normals

This tool translates selected vertices/edges/faces along their own normal (perpendicular to the face), which, on “standard normal meshes”, will shrink/fatten them.

This transform tool does not take into account the pivot point or transform orientation.


mesh before shrink/flatten


Inflated using a positive value


Shrunk using a negative value

Smooth

Mode: Edit mode

Panel: Mesh Tools (Editing context, F9)

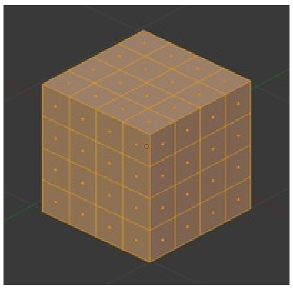
Hotkey: CtrlV » Smooth vertex

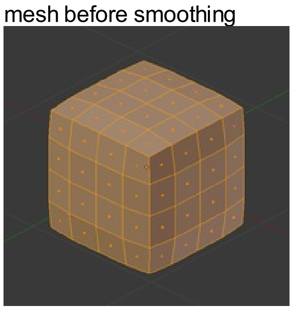
Menu: Mesh » Vertices » Smooth vertex

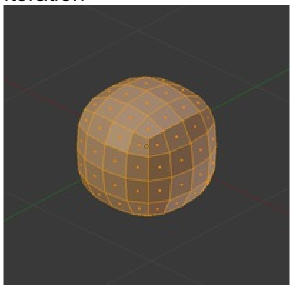
This tool smooths the selected components by averaging the angles between faces. After using the tool, options appear in the Tool Shelf:

Number of times to smooth
: The number of smoothing iterations

Axes
: Limit the effect to certain axes.


mesh before smoothing


mesh after 1 smoothing iteration


mesh after 10 smoothing iterations

Laplacian Smooth

Mode: Edit mode

Hotkey: W » Laplacian Smooth

See the Laplacian Smooth Modifier for details.

Laplacian smooth uses an alternative smoothing algorithm that better preserves the overall mesh shape. Laplacian smooth exists as a mesh operation and as a non-destructive modifier.

Note
The Smooth modifier, which can be limited to a Vertex Group, is a non-destructive alternative to the smooth tool.

Real Smoothing versus Shading Smoothing
Do not mistake this tool with the shading smoothing options described at this page, they do not work the same! This tool modifies the mesh itself, to reduce its sharpness, whereas Set Smooth/AutoSmooth and co. only control the way the mesh is shaded, creating an *illusion* of softness – but without modifying the mesh at all…

Noise

Mode: Edit mode

Panel: Mesh tools panel (Editing context)

Note
Noise is an old feature. The Displace Modifier is a non-destructive alternative to the Noise tool and is a more flexible way to realize these sort of effects. The key advantages of the modifier are that it can be canceled at any moment, you can precisely control how much and in which direction the displacement is applied, and much more….
See also the ANT Landscape add-on.

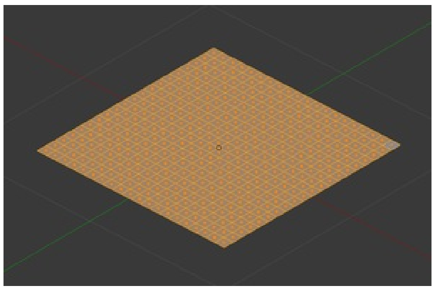
The Noise function allows you to displace vertices in a mesh based on the grey values of the first texture slot of the material applied to the mesh.

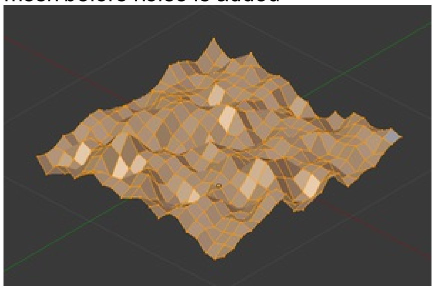
The mesh must have a material and a texture assigned to it for this tool to work. To avoid having the texture affect the material's properties, it can be disabled in the texture menu.

The Noise function displaces vertices along the object’s ±Z-Axis only.

Noise permanently modifies your mesh according to the material texture. Each click adds onto the current mesh. For a temporary effect, map the texture to Displacement for a render-time effect. In Object/Edit mode, your object will appear normal, but will render deformed.

The deformation can be controlled by modifying the Mapping panel and/or the texture's own panel (e.g. Clouds, Marble, etc.).


mesh before noise is added


mesh after noise is added, using basic cloud texture

Mesh Duplicating Tools

This section covers mesh editing tools that add additional geometry by duplicating existing geometry in some way.

- Duplicate Geometry.
- Extrusion.
- Spin.
- Screw.

Multiple Viewports
When you use one of the duplication tools in the Mesh Tools panel, Blender cannot guess which view you want to work in – if you have more than one opened, of course… As the view is often important for these tools, once you have activated one, your cursor turns into a sort of question mark – click with it inside the window you want to use.

Duplicate

Mode: Edit mode

Hotkey: ⇧ ShiftD

Menu: Mesh » Duplicate

This tool simply duplicates the selected elements, without creating any links with the rest of the mesh (unlike extrude, for example), and places the duplicate at the location of the original. Once the duplication is done, *only the newduplicated elements are selected*, and you are automatically placed in grab/move mode, so you can translate your copy elsewhere…

In the Tool Shelf are settings for Vector offset, Proportional Editing, Duplication Mode (non-functional?), and Axis Constraints.

Note that duplicated elements belong to the same vertex groups as the “original” ones. The same goes for the material indices, the edge’s Sharp and Seam flags, and probably for the other vertex/edge/face properties…

Extrude

Extrude Region

Mode: Edit mode

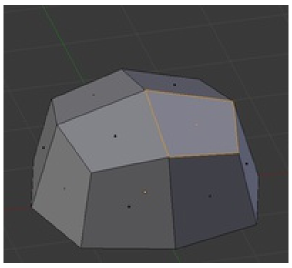
Panel: Mesh Tools » Extrude

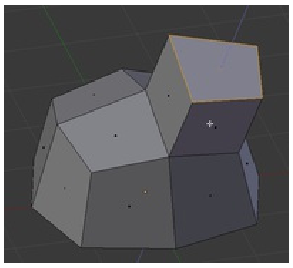
Hotkey: E or AltE

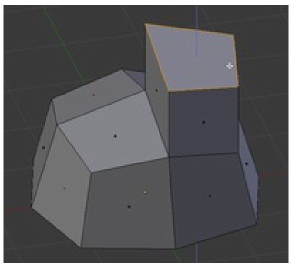
Menu: Mesh » Extrude Region

One tool of paramount importance for working with meshes is the Extrude tool. It allows you to create parallelepipeds from rectangles and cylinders from circles, as well as easily create such things as tree limbs. Extrude is one of the most frequently used modeling tools in Blender. It's simple, straightforward, and easy to use, yet very powerful.

The selection is extruded along the common normal of selected faces. In every other case the extrusion can be limited to a single axis by specifying an axis (e.g. X to limit to the X axis or ⇧ ShiftX to the YZ plane. When extruding along the face normal, limiting movement to the global Z axis requires pressing Z twice, once to disable the face normal Z axis limit, and once to enable the global Z axis limit.


Selected face


During extrude


Set to Z axis

Although the process is quite intuitive, the principles behind Extrude are fairly elaborate as discussed below:

- First, the algorithm determines the outside edge-loop of the extrude; that is, which among the selected edges will be changed into faces. By default (see below), the algorithm considers edges belonging to two or more selected faces as internal, and hence not part of the loop.
- The edges in the edge-loop are then changed into faces.
- If the edges in the edge-loop belong to only one face in the complete mesh, then all of the selected faces are duplicated and linked to the newly created faces. For example, rectangles will result in parallelepipeds during this stage.
- In other cases, the selected faces are linked to the newly created faces but not duplicated. This prevents undesired faces from being retained "inside" the resulting mesh. This distinction is extremely important since it ensures the construction of consistently coherent, closed volumes at all times when using Extrude.
- When extruding completely closed volumes (like e.g. a cube with all its six faces), extrusion results merely in a duplication, as the volume is duplicated, without any link to the original one.
- Edges not belonging to selected faces, which form an "open" edge-loop, are duplicated and a new face is created between the new edge and the original one.
- Single selected vertices which do not belong to selected edges are duplicated and a new edge is created between the two.

Extrude Individual

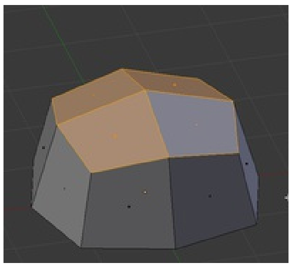
Mode: Edit mode

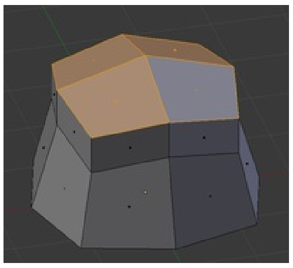
Panel: Mesh Tools » Extrude Individual

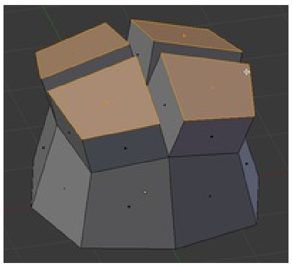
Hotkey: AltE

Menu: Mesh » Extrude Individual

Extrude Individual allows you to extrude a selection of multiple faces as individuals, instead of as a region. The faces are extruded along their own normals, rather than their average. This has several consequences: first, "internal" edges (i.e. edges between two selected faces) are no longer deleted (the original faces are).


Selection of multiple faces


Extruded using extrude region

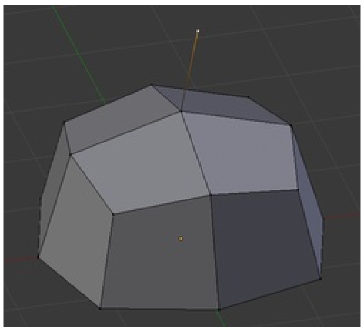

Extruded using Extrude Individual

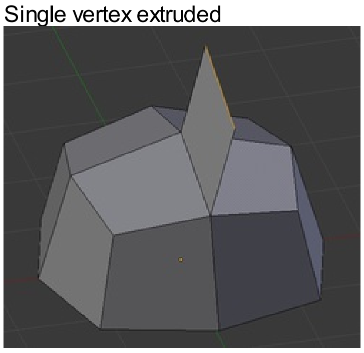
Extrude Edges and Vertices Only

Mode: Edit mode, Vertex and Edge

Hotkey: AltE

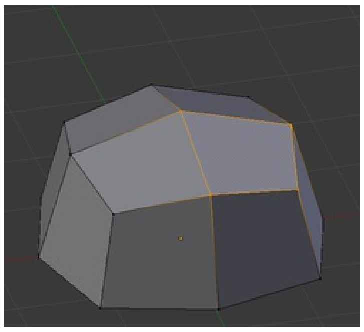
If vertices are selected while doing an extrude, but they do not form an edge or face, they will extrude as expected, forming a non-manifold edge. Similarly, if edges are selected that do not form a face, they will extrude to form a face.

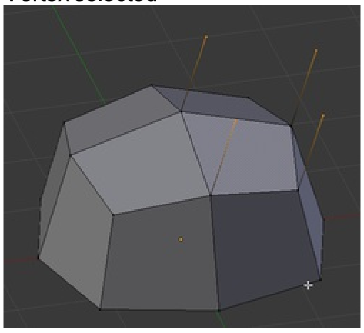

Single vertex extruded

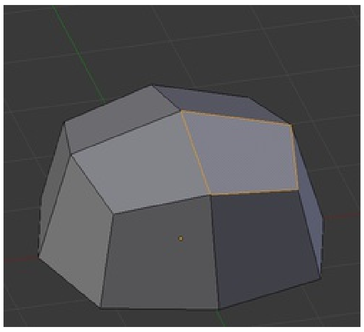

Single edge extruded

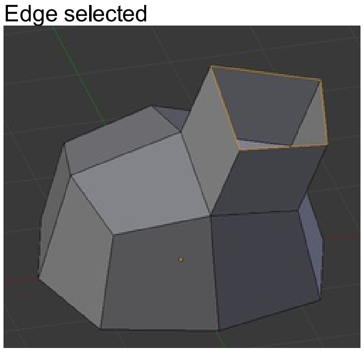
When a selection of vertices forms an edge or face, it will extrude as if the edge was selected. Likewise for edges that form a face.

To force a vertex or edge selection to extrude as a vertex or edge, respectively, use AltE to access the Extrude Edges Only and Vertices Only.


Vertex selected


Vertices Only extrude


Edge selected


Edge Only extrude

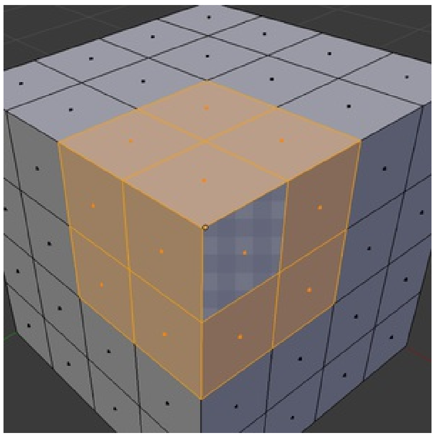
Inset

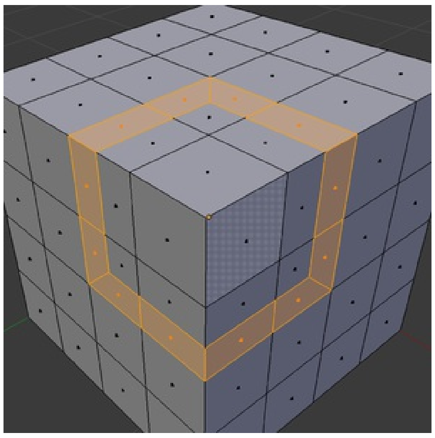
Mode: Edit mode

Hotkey: I

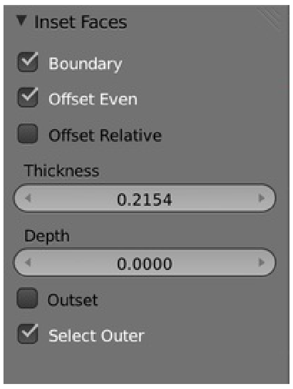
Menu: Mesh » Faces » Inset or CtrlF » Inset

This tool takes the currently selected faces and creates an inset of them, with adjustable thickness and depth. The tool is modal, such that when you activate it, you may adjust the thickness with your mouse position. You may also adjust the depth of the inset during the modal operation by holding Ctrl.


Selection to inset


Selection with inset

Options


▼ Inset Faces

✓ Boundary

✓ Offset Even

Offset Relative

Thickness 0.2154

Depth 0.0000

Outset

✓ Select Outer

Inset Operator Settings

Boundary
: Determines whether open edges will be inset or not.

Offset Even
: Scale the offset to give more even thickness.

Offset Relative
: Scale the offset by surrounding geometry.

Thickness
: Set the size of the inset.

Depth
: Raise or lower the newly inset faces to add depth.

Outset
: Create an outset rather than an inset.

Select Outer
: Toggle which side of the inset is selected after operation.

Spin

Mode: Edit mode

Panel: Mesh Tools (Editing context)

Use the Spin tool to create the sort of objects that you would produce on a lathe (this tool is often called a “lathe”-tool or a “sweep”-tool in the literature, for this reason). In fact, it does a sort of circular extrusion of your selected elements, centered on the 3D cursor, and around the axis perpendicular to the working view…

- The point of view will determine around which axis the extrusion spins…
- The position of the 3D cursor will be the center of the rotation.

Here are its settings:

Steps
: Specifies how many copies will be extruded along the “sweep”.

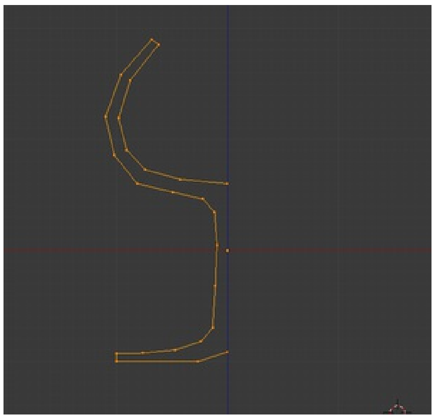
Dupli
: When enabled, will keep the original selected elements as separated islands in the mesh (i.e. unlinked to the result of the spin extrusion).

Angle
: specifies the angle “swept” by this tool, in degrees (e.g. set it to **180 °** for half a turn).

Center
: Specifies the center of the spin. By default it uses the cursor position.

Axis
: Specify the spin axis as a vector. By default it uses the view axis.

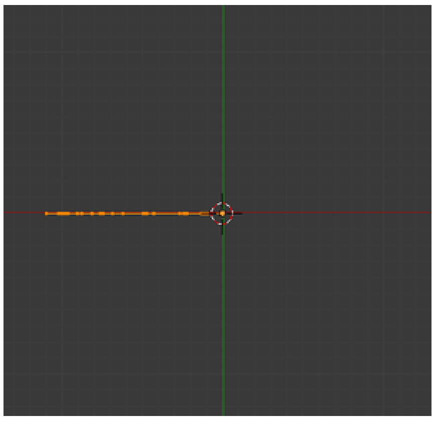
Example


Glass profile.

First, create a mesh representing the profile of your object. If you are modeling a hollow object, it is a good idea to thicken the outline. (*Glass profile*) shows the profile for a wine glass we will model as a demonstration.

Go to the Edit mode and select all the vertices of the Profile with A.

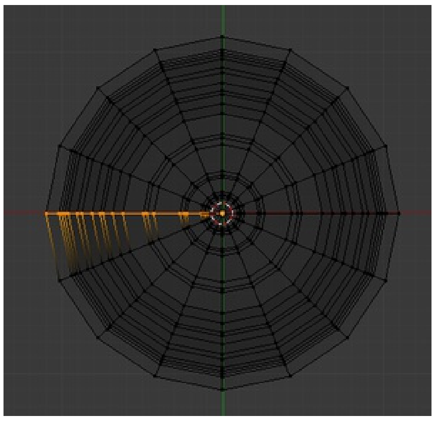
We will be rotating the object around the cursor in the top view, so switch to the top view with 7 NumPad.

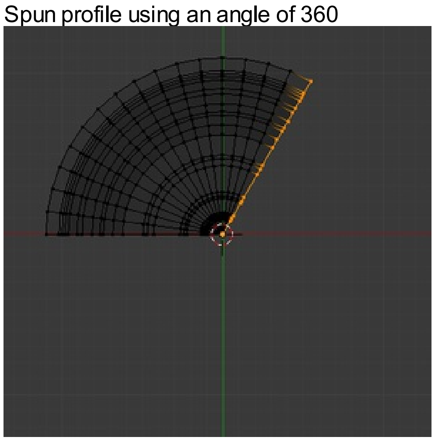

Glass profile, top view in Edit mode, just before spinning.

Place the cursor along the center of the profile by selecting one of the vertices along the center, and snapping the 3D cursor to that location with ⇧ ShiftS » Cursor -> Selection. (*Glass profile, top viewin Edit mode, just before spinning*) shows the wine glass profile from top view, with the cursor correctly positioned.

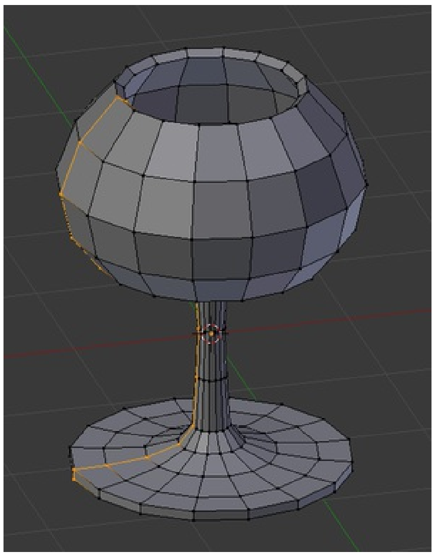
Click the Spin button. If you have more than one 3D view open, the cursor will change to an arrow with a question mark and you will have to click in the window containing the top view before continuing. If you have only one 3D view open, the spin will happen immediately. (*Spun profile*) shows the result of a successful spin.

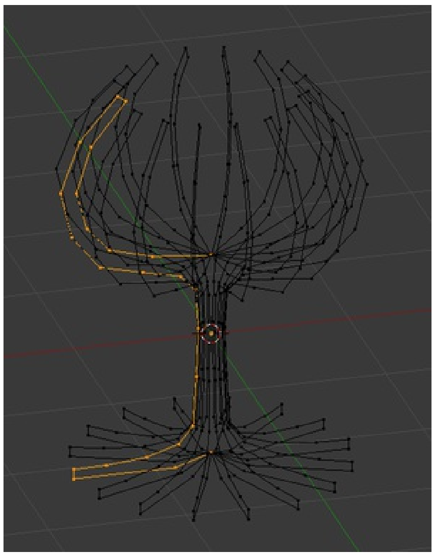
Angle


Spun profile using an angle of 360

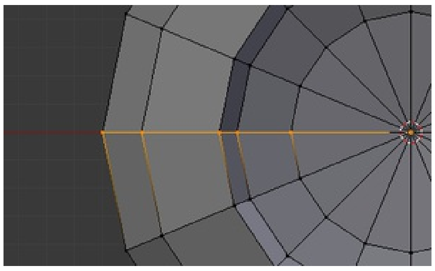

Spun profile using an angle of 120

Dupli


Result of spin operation


Result of Dupli enabled

Merge Duplicates


Duplicate vertices

The spin operation leaves duplicate vertices along the profile. You can select all vertices at the seam with Box select (B) shown in (*Seam vertex selection*) and perform a Remove Doubles operation.

Notice the selected vertex count before and after the Remove Doubles operation (*Vertex count after removing doubles*). If all goes well, the final vertex count (38 in this example) should match the number of the original profile noted in (*Mesh data – Vertex and face numbers*). If not, some vertices were missed and you will need to weld them manually. Or, worse, too many vertices will have been merged.

Merging two vertices in one
To merge (weld) two vertices together, select both of them by ⇧ Shift RMB clicking on them. Press S to start scaling and hold down Ctrl while scaling to scale the points down to 0 units in the X, Y and Z axis. LMB to complete the scaling operation and click the Remove Doubles button in the Buttons window, Editing context (also available with W » Remove Doubles). Alternatively, you can use W » Merge from the same Specials menu (or AltM). Then, in the new pop-up menu, choose whether the merged vertex will be at the center of the selected vertices or at the 3D cursor. The first choice is better in our case!

Recalculate Normals

All that remains now is to recalculate the normals to the outside by selecting all vertices, pressing CtrlN and validating Recalc Normals Outside in the pop-up menu.

Screw Tool

Mode: Edit Mode

Panel: Edit Mode → Mesh Tools (shortcut T) → Add → Screw Button

Introduction

The Screw Tool is one of the most used tools for generating continuous circular profiles in Blender since its ancient versions because the generated profiles are very predictable, light to deal with and well connected. This tool helps artists generate clean circular-profile meshes.

You can see some examples of Meshes generated with the Screw tool in Fig. 1 – Wood Screw tip done with the screw tool and Fig. 2 – Spring done with the screw tool.

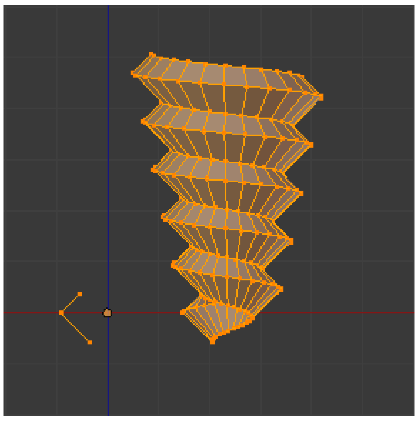


Fig. 1 - Wood Screw tip done with the screw tool

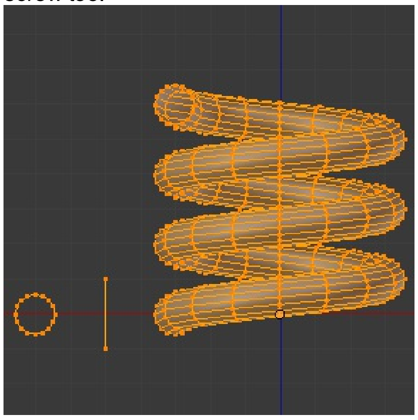


Fig. 2- Spring done with the screw tool

Description

The Screw tool combines a repetitive Spin with a translation, to generate a screw-like, or spiral-shaped, object. Use this tool to create screws, springs, or shell-shaped structures (Sea shells, Wood Screw Tips, Special profiles, etc).

The main difference between the Screw Tool and the Screw Modifier is that the Screw Tool can calculate the angular progressions using the basic profile angle automatically or adjusting the Axis angular vector without using a second modifier (for example, using the Screw Modifier with a Bevel Modifier, Curve Modifier, etc...), resulting in a much cleaner approach for vertex distribution and usage.

This tool works using open or closed profiles, as well as profiles closed with faces. You can use profiles like an open-edge part that is a part of a complete piece, as well as a closed circle or a half-cut sphere, which will also close the profile end.

Usage

- This tool works only with Meshes.
- In Edit Mode, the button for the Screw tool operation is located in the Mesh Tools Panel, (shortcut T) → Add → Screw Button.
- To use this tool, you need to create at least one open profile or line to be used as a vector for the height, angular vector and to give Blender a direction.
- The Screw function uses two points given by the open line to create an initial vector to calculate the height and basic angle of the translation vector that is added to the “Spin” for each full rotation (see examples below). If the vector is created with only two vertices at the same **X**, **Y** and **Z** location (which won't give Blender a vector value for height), this will create a normal “Spin”.
- Having at least one vector line, you can add other closed support profiles that will follow this vector during the extrusions (See limitations).
- The direction of the extrusions is calculated by two determinant factors, your point of view in Global Space and the position of your cursor in the 3DView Space using Global coordinates.
- The profile and the vector must be fully selected in Edit Mode before you click the Screw Button (See Limitations.)

- When you have the vector for the open profile and the other closed profiles selected, click the Screw Button.

Limitations

There are strict conditions about your profile selection when you want to use this tool. You must have at least one open line or open profile, giving Blender the starting Vector for extrusion, angular vector and height. (e.g. a simple edge, a half circle, etc…). You need only to ensure that at least one reference line has two “free” ends. If two open Lines are given, Blender won't determine which of them is the vector, and will then show you an error message, “`You have to select a string of connected vertices too`”. You need to select all of the profile vertices that will participate in the Screw Tool operation; if they are not properly selected, Blender will also show you the same message.

Note that the open line is always extruded, so if you only use it to “guide” the screw, you will have to delete it after the tool completion (use linked-selection, CtrlL, to select the whole extrusion of the open line).

If there is any problem with the selection or profiles, the tool will warn you with the error message: “`You have to select a string of connected vertices too`” as seen in Fig. 3 and 4, both in the info Window and at the place where you clicked to start performing the operation (when you click the Screw Button).

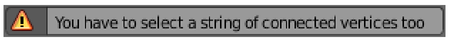
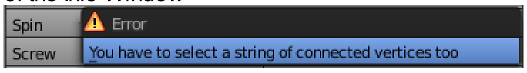

You have to select a string of connected vertices too

Fig. 3 - Screw Error message in the Header of the Info Window



You have to select a string of connected vertices too

Fig. 4 - Error message when clicking in the Screw Tool with an incorrect or bad selection

You may have as many profiles as you like (like circles, squares, and so on) – Note that not all vertices in a profile need to be in the same plane, even if this is the most common case. You may also have other, more complex, selected closed islands, but they have to be closed profiles because Blender will seek for only one open profile for the translation, height and angular vector. Some closed meshes that overlap themselves may not screw correctly (for example: Half UVsphere = OK, more than half = could cause the Screw Tool to have wrong behavior or errors), and profiles that are closed with faces (like a cone or half sphere) will be closed automatically at their ends, like if you were extruding a region.

Simple way to not result in error

Only one open Profile, all of the others can be closed, avoid volumes and some profiles closed with faces...

Options

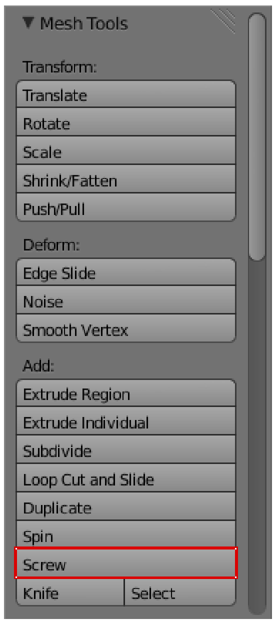


Fig. 5 - Screw Tool in the Mesh Tools Panel (Edit Mode)

This tool is an interactive and modal tool, and only works in the Edit Mode.

Once you click in the Screw tool in the Mesh Tools Panel, Blender will enter in the Screw interactive mode, and the Operator Panel at the end of the Mesh Tools Panel will be replaced so you can adjust the values explained below. To show the Mesh Tools Panel, use the shortcut T in the Edit Mode of the 3D View Window.

(See Fig. 5 - Screw Tool in the Mesh Tools Panel (Edit Mode), red box)

Once you perform any other operation, Blender leaves the interactive mode and accepts all of the values. Because it's modal, you can't return to the interactive mode after completing/leaving the operation or changing from Edit Mode to Object Mode. If you want to restart the operation from its beginning, you can hit CtrlZ at any time in Edit Mode.

- The basic location of the cursor at the point of view (using Global coordinates) will determine around which axis the selection is extruded and spun at first (See Fig. 6 - Cursor Basic Location - Transform Panel). Blender will copy your cursor location coordinates to the values present in the Center values of the Screw interactive Panel. Depending on the Global View position, Blender will automatically add a value of **1** to one of the Axis Vectors, giving the profiles a starting direction for the Screw Operation and also giving a direction for the extrusions. (See examples below.)
- The position of the 3D cursor will be the starting center of the rotation. Subsequent operations (e.g. pressing the Screw button again), will start from the last selected element. Continuous operations without changing the selection will repeat the operation continuously from the last point.

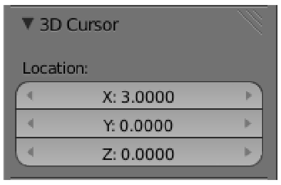


Fig. 6 - Cursor Basic Location - Transform Panel

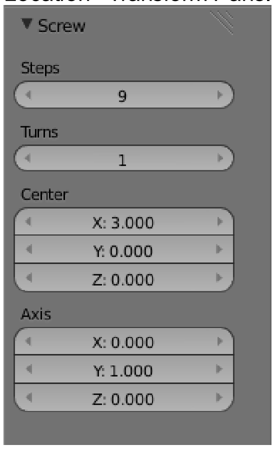


Fig. 7 - Screw Interactive Panel - Mesh Tools Panel (Edit Mode)

Center
: These numeric fields specify the center of the spin. When the tool is called for the first time, it will copy the **X**, **Y** and **Z** location (Global Coordinates) of the cursor presently in the 3D View to start the operation. You can specify the cursor coordinates using the Transform Panel in 3D View, using shortcut T to toggle the Panel, and typing in the 3D Cursor Location coordinates, but, unlike in previous Blender Versions (prior to 2.5x), now you can adjust those coordinates interactively and specify another place for the spin center during the interactive session. (See Fig. 7 - Screw Interactive Panel - Mesh Tools Panel (Edit Mode))

Steps
: This numeric field specifies how many extrusion(s) will be done for each **360°** turn. The steps are evenly distributed by dividing **360º** by the number of steps given. The minimum value is **3**; the maximum is **256** (See Fig. 7)

Turns
: This numeric field specifies how many turns will be executed. Blender will add a new full **360°** turn for each incremental number specified here. The minimum value is **1**; the maximum is **256**. (See Fig. 7)

Axis
: These **3** numeric fields vary from **-1.0** to **1.0** and are clamped above those limits. These values correspond to angular vectors from **-90** to **90** degrees. Depending on the position where you started your cursor location and Object operation in the viewport and its axis positions in Global View space and coordinates, Blender will give the proper Axis vector a value of **1**, giving the angular vector of the profile a starting direction and giving the extrusions a starting direction based on your view. Blender will let you adjust your axis angular vectors and you can tweak your object such that you can revert the direction of the screw operation (by reverting the angular vector of the height), meaning you can revert the clockwise and counterclockwise direction of some operations, and also adjust the angular vectors of your profile, bending it accordingly. (See Fig. 7)

Examples

The Spring example

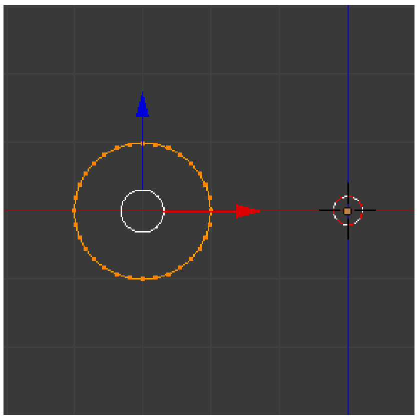


Fig. 8 - Circle placed at X -3,0,0

- Open Blender and delete the default Cube.
- Change from perspective to orthographic view using shortcut Numpad5.
- Change your view from *User Ortho* to *Front Ortho*, using the shortcut Numpad1. You will see the X (red) and Z (blue) coordinate lines.
- In case you have moved your cursor by clicking anywhere in the screen, again place your cursor at the Center, using the shortcut ⇧ ShiftS choosing *Cursor to Center* or the Transform Panel, placing your cursor at (**0,0,0**) typing directly into the Cursor 3D Location.
- Add a circle using shortcut ⇧ ShiftA and choosing → Mesh → Circle.
- Rotate this circle using the shortcut RX and typing **90** and ↵ Enter.
- Apply the Rotation using CtrlA and choosing *Rotation*
- Grab and move this circle to the left **3** Blender Units on the **X** Axis; you can use the shortcut Ctrl while grabbing with the mouse using the standard transform widgets (clicking on the red arrow shown with the object and grabbing while using shortcut Ctrl until the down left info in the 3D View marks **D. -3.0000 (3.0000) Global**), or press the shortcut GX and typing **-3** and ↵ Enter. You can use the Transform Panel (toggled with the shortcut T , and type **-3** and ↵ Enter in the Location too. (See the Fig. 8 - Circle placed at X -3,0,0).
- You will have to scale your circle using the shortcut S and typing **.5**, then ↵ Enter.
- Now enter Edit Mode using shortcut ⇆ Tab.
- De-select all vertices using the shortcut A.

Now we will create a height vector for Blender:

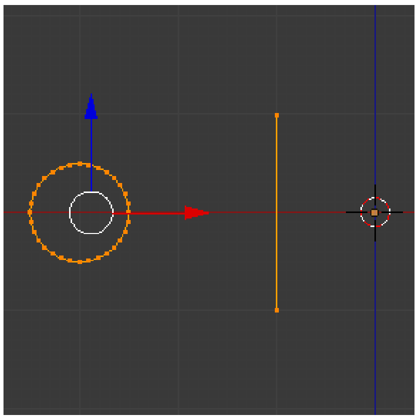


Fig. 9 - Profile and vector created

- Press Ctrl and Left click LMB near the circle, in more or less at the light grey line of the square above the circle, and, while still pressing Ctrl, Left Click LMB again in the grey line below the circle. You have created two vertices and an Edge, which Blender will use as the first height and angle vector.
- Now, in the Transform Panel, in the median, clicking in the Global coordinates, for the **X**, **Y**, and **Z** coordinates, put **(-2, 0, -1)**.
- Right Click RMB in the other vertex, and again, type its coordinates for **X**, **Y** and **Z** to **(-2, 0, 1)**. This will create a straight vertical line with 2 Blender units of Height.
- De-select and select everything again with the shortcut A. (See Fig. 9 - Profile and vector created)
- Place again your cursor at the center. (Repeat step 2)
- At this point, we will save this Blender file to recycle the Spring for another exercise; click with LMB in *File*, it is placed at the header of the Info Window, (At the top left side), and choose *Save as*. Our suggestion is to name it *Screw Spring Example.blend* and click in *Save as Blender file*. You can also use the shortcut ⇧ ShiftCtrlS to open the File Browser Window in order to save your Blender file.
- Click Screw and adjust the Steps and Turns as you like and we have a nice spring, but now here comes the interesting part!

Clockwise and Counterclockwise using the Spring Example

Still in the interactive session of the Screw Tool, you will see that the **Z** Axis Value of the Screw Panel is set to **1.000**. Left click LMB in the middle of the Value and set this value to **-1.000**. At first, the Spring was being constructed in a Counterclockwise direction, and you reverted the operation **180** degrees in the **Z** Axis. This is because you have changed the angular vector of the height you have given to Blender to the opposite direction (remember, **-90** to **90** = **180** degrees ?). See Fig. 10 - Counterclockwise direction and Fig. 11 - Flipped to Clockwise direction.

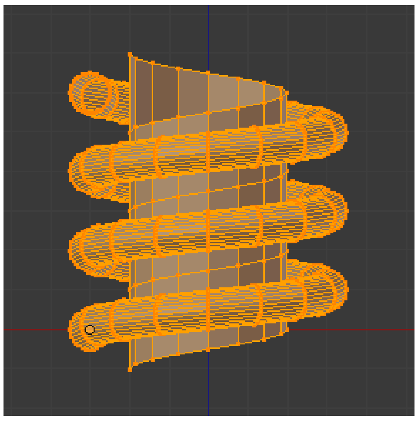


Fig. 10 - Counterclockwise direction

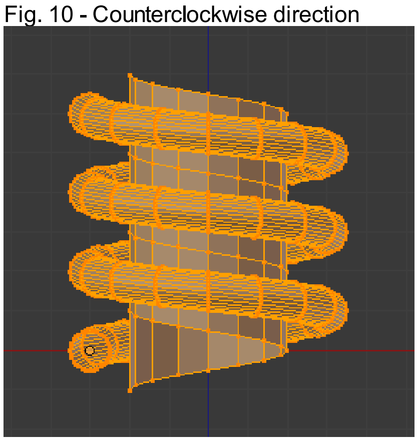


Fig. 11 - Flipped to Clockwise direction.

It's also important to note that this vector is related to the same height vector axis used for the extrusion and we have created a parallel line with the **Z** Axis, so, the sensibility of this vector is in practical sense reactive only to negative and positive values because it's aligned with the extrusion axis. Blender will clamp the positive and negative to its maximum values to make the extrusion follow a direction, even if the profile starts reverted. The same rule applies to other Global axes when creating the Object for the Screw Tool; this means if you create your Object using the Top View (Shortcut Numpad7 with a straight parallel line following another axis (for the Top View, the **Y Axis**), the vector that gives the height for extrusion will also change abruptly from negative to positive and vice versa to give the extrusion a direction, and you will have to tweak the corresponding Axis accordingly to achieve the Clockwise and Counterclockwise effect.

Vectors that aren't parallel with Blender Axis
The high sensibility for the vector doesn't apply to vectors that give the Screw Tool a starting angle (Ex: any non-parallel vector), meaning Blender won't need to clamp the values to stabilize a direction for the extrusion, as the inclination of the vector will be clear for Blender and you will have the full degree of freedom to change the vectors. Our example is important because it only changes the direction of the profile without the tilt and/or bending effect, as there is only one direction for the extrusion, parallel to one of the Blender Axes

Bending the Profiles using the Spring Example

Still using the Spring Example, we can change the remaining vector for the angles that aren't related to the extrusion Axis of our Spring, thus bending our spring with the remaining vectors and creating a profile that will also open and/or close because of the change in starting angular vector values. What we are really doing is changing the starting angle of the profile prior to the extrusions. It means that Blender will connect each of the circles inclined with the vector you have given. Below we show two bent Meshes using the Axis vectors and the Spring example. See Fig. 12 and Fig. 13. These two Meshes generated with the Screw tool were created using the Top Ortho View.

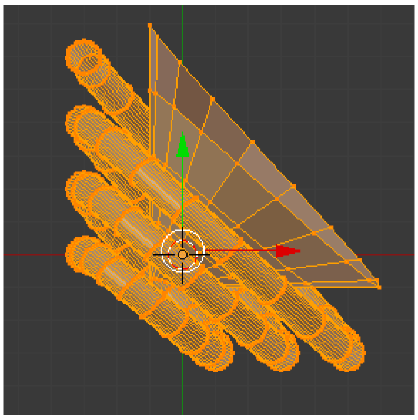


Fig. 12 - Bended Mesh, Example 1 - The Axis will give the profile a starting vector angle

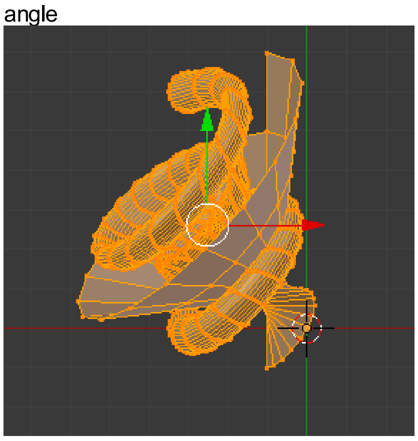


Fig. 13 - Bended Mesh Example 2 - The vector angle is maintained along the extrusions

Creating perfect Screw Spindles

Using the Spring Example, it's easy to create perfect Screw Spindles (like the ones present in normal screws that we can buy in hardware stores). Perfect Screw Spindles use a profile with the same height as its vector, and the beginning and ending vertex of the profile are placed at a straight parallel line with the axis of extrusion. The easiest way of achieving this effect is to create a simple profile where the beginning and ending vertices create a straight parallel line. Blender won't take into account any of the vertices present in the middle but those two to take its angular vector, so the spindles of the screw (which are defined by the turns value) will assembly perfectly with each other.

- Open Blender and click in *File* located at the header of the Info Window again, choose *Open Recent* and the file we saved for this exercise. All of the things will be placed exactly the way you saved before. Choose the last saved Blender file; in the last exercise, we gave it the name *ScrewSpring Example.blend*.
- Press the shortcut A to de-select all vertices.
- Press the shortcut B, and Blender will change the cursor; you're now in border selection mode.
- Open a box that selects all of the circle vertices except the two vertices we used to create the height of the extrusions in the last example.
- Use the shortcut X to delete them.
- Press the shortcut A to select the remaining vertices.
- Press the shortcut W for the *Specials Menu*, and select *Subdivide*
- Now, click with the Right Mouse button at the middle vertex.
- Grab this vertex using the shortcut GX, type **-1** and ↵ Enter. See Fig. 14 - Profile for a perfect screw spindle.
- At this point, we will save this Blender file to recycle the generated Screw for another exercise; click with LMB in *File*-- it is in the header of the Info Window (at the top left side), and choose *Save as*. Our suggestion is to name it *ScrewHardware Example.blend* and click in *Save as Blender file*. You can also use the shortcut ⇧ ShiftCtrlS to open the File Browser Window in order to save your Blender file.
- Press shortcut A twice to de-select and select all vertices again.
- Now press Screw.
- Change Steps and Turns as you like. Fig. 15 - Generated Mesh - Shows you an example of the results.

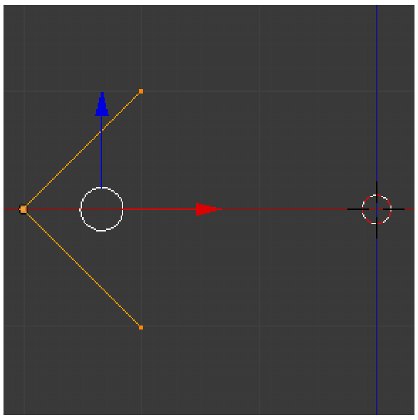


Fig. 14 - Profile for a perfect screw spindle. The starting and ending vertices are forming a parallel line with the Blender Axis

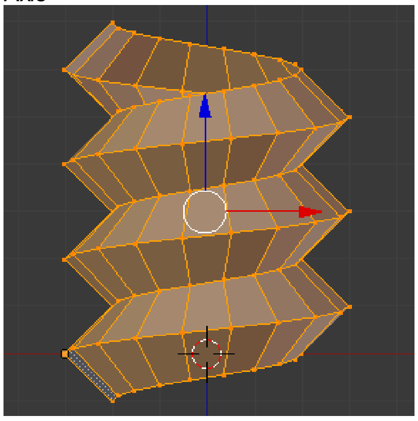


Fig. 15 - Generated Mesh. You can use this technique to perform normal screw modeling.

Here, in Fig. 16 and Fig. 17, we show you an example using a different profile, but maintaining the beginning and ending vertices at the same position. The generated mesh looks like a medieval ramp!

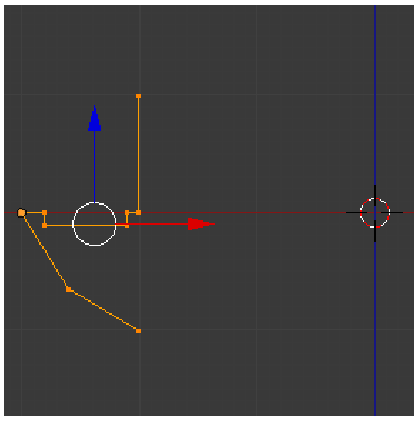


Fig. 16 - Profile with starting and ending vertices forming a parallel line with the Blender Axis

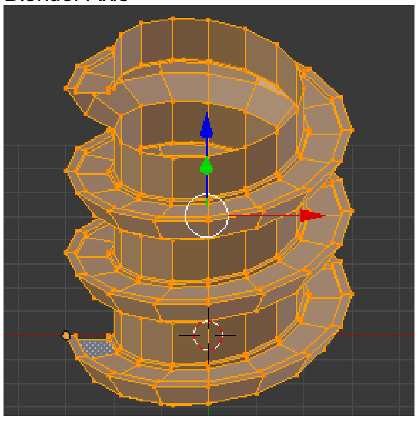


Fig. 17 - Generated Mesh with the profile

at the left. We have inclined the visualization a bit.

As you can see, the Screw spindles are perfectly assembled with each other, and they follow a straight line from top to bottom. You can also change the Clockwise and Counterclockwise direction using this example, to create right and left screw spindles. At this point, you can give the screw another dimension, changing the Center of the Spin Extrusion, making it more suitable to your needs or calculating a perfect screw and merging its vertices with a cylinder, modeling its head, etc.

A Screw Tip

As we have explained before, the Screw tool generates clean and simple meshes to deal with; they are light, well-connected and are created with very predictable results. This is due to the Blender calculations taking into account not only the height of the vector, but also its starting angle. It means that Blender will connect the vertices with each other in a way that they follow a continuous cycle along the extruded generated profile.

In this example, you will learn how to create a simple Screw Tip (like the ones we use for wood; we have shown an example at the beginning of this page). To make this new example as short as possible, we will recycle our last example (again).

- Open Blender and click in *File* located in the header of the Info Window again; choose *Open Recent* and the file we saved for this exercise. All of the things will be placed exactly the way you saved before. Choose the last saved Blender file; in the last exercise, we gave it the name *ScrewHardware Example.blend*.
- Grab the upper vertex and move a bit to the left, but no more than you have moved your last vertex. (See Fig. 18 - Profile With Starting Vector Angle)
- Press the shortcut A twice to de-select and select all.
- Press the shortcut ⇧ ShiftS and select *Cursor to Center*
- Press Screw.

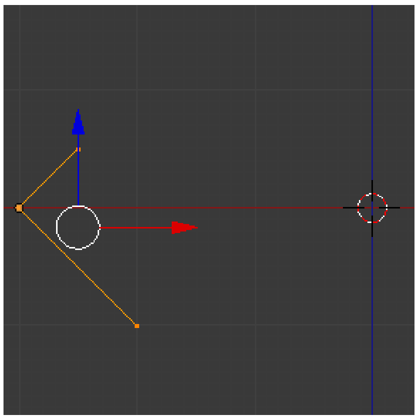


Fig. 18 - Profile With Starting Vector Angle

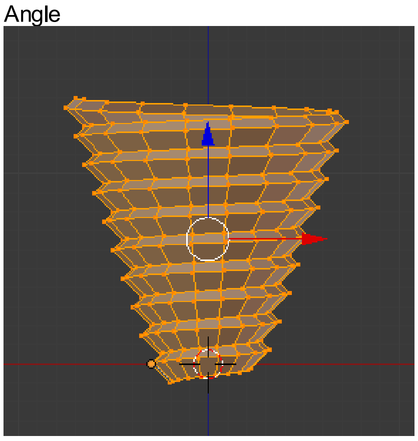


Fig. 19 - Generated Mesh with the Profile

As you can see in Fig. 19, Blender follows the basic angular vector of the profile, and the profile basic angle determines whether the extruded subsequent configured turns will open or close the resulting mesh following this angle. The vector of the extrusion angle is determined by the starting and ending Vertex of the profile.

Screw Tool - Evolution since 2.5x

During the recode of Blender, from 2.4x to 2.5x series, the screw tool received lots of improvements. In Blender 2.4x series, the screw tool uses only one cursor position for its axis reference at a time, meaning you cannot tweak your object center changing the reference position without restarting the operation from its beginning. In 2.4x series, you also cannot change the starting angular vector, the only one available was the vector that gives the clockwise and counterclockwise direction, and the angular vector of the tool during the screw operation couldn't be adjusted.

In 2.5x and above, you can not only change the reference position after it's chosen with the mouse cursor, using the interactive panel in the Mesh Tools in the Edit Mode (Shortcut T) to change the center during the interactive session, but you can also change the angular vector of the generated object by adjusts during the interactive session.

Another difference is that the clockwise and counterclockwise rotation of the Screw Tool now is determined by the axis vector tweak, meaning that you can change the direction of the screw rotation adjusting the corresponding **X**, **Y** and **Z** vector axis in positive and negative directions, it will depend on the orientation you have placed your object for creation and its center coordinates for spin and extrusion, we will explain the most common case in the examples of this page.

Blender will also determine automatically, depending on your view, the proper direction for the extrusion axis, meaning that you can change, at any time, the screw extrusion direction changing the global view alignment.

The rotation axis, still passing through the 3D cursor, is now free, but it's still preferable to align it with the y-axis of the view (i.e. up-down on the screen). So, the best way to start using this tool is to align your view with the front orthographic view using Numpad1 to create the Global height of the extrusions aligned with the Local Axis of your object. Blender will determine automatically your extrusion axis when you align your Vector with one of the Blender Global Axis, giving the proper axis vector a value of **1**.

Mesh Subdividing Tools

Subdividing adds resolution by cutting existing faces and edges into smaller pieces. There are several tools that allow you to do this:

Subdivide	Divide a face or edge into smaller units, adding resolution.
Loop Subdivide	Insert a loop of edges between existing ones
Vertex Connect	Connects selected vertices with edges that split faces.
Knife Subdivide	Cut edges and faces interactively
Bevel	Subdivides edges or vertices, making them faceted or rounded

Subdivide

Mode: Edit mode

Panel: Mesh Tools (Editing context)

Hotkey: W » 1 NumPad/2 NumPad

Menu: Mesh » Edges » Subdivide, Specials » Subdivide/Subdivide Smooth

Subdividing splits selected edges and faces by cutting them in half or more, adding necessary vertices, and subdividing accordingly the faces involved, following a few rules, depending on the settings:

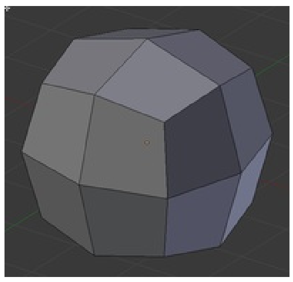
- When only one edge of a face is selected (Tri mode), triangles are subdivided into two triangles, and quads, into three triangles.
- When two edges of a face are selected:
  - If the face is a triangle, a new edge is created between the two new vertices, subdividing the triangle in a triangle and a quad.
  - If the face is a quad, and the edges are neighbors, we have **three** possible behaviors, depending on the setting of Corner Cut Type (the drop-down menu next to the Subdivide button, in Mesh Tools panel) See below for details.
  - If the face is a quad, and the edges are opposite, the quad is just subdivided in two quads by the edge linking the two new vertices.
- When three edges of a face are selected:
  - If the face is a triangle, this means the whole face is selected – it is then sub-divided in four smaller triangles.
  - If the face is a quad, first the two opposite edges are subdivided as described above. Then, the “middle” edge is subdivided, affecting its new “sub-quad” as described above for only one edge.
- When four edges of a face (a quad) are selected, the face is subdivided into four smaller quads.

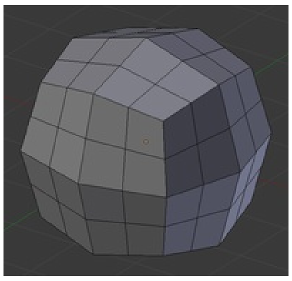
Options

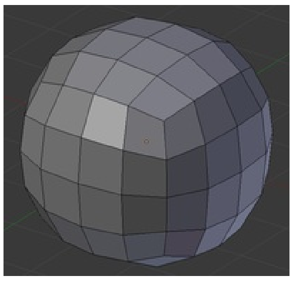
These options are available in the Tool Panel after running the tool;

Number of Cuts
: Specifies the number of cuts per edge to make. By default this is 1, cutting edges in half. A value of 2 will cut it into thirds, and so on.

Smoothness
: Displaces subdivisions to maintain approximate curvature, The effect is similar to the way the subdivision modifier might deform the mesh.


Mesh before subdividing


Subdivided with no smoothing


Subdivided with smoothing of 1

Quad/Tri Mode
: Forces subdivide to create triangles instead of ngons, simulating old behavior (see examples below)

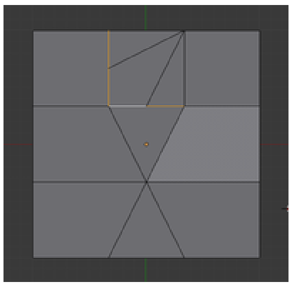
Corner Cut Type
: This drop-down menu controls the way quads with only two adjacent selected edges are subdivided

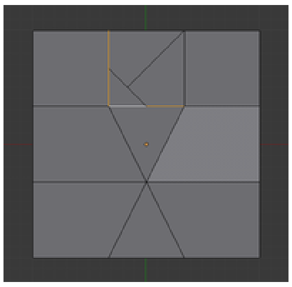
  Fan
  : the quad is sub-divided in a fan of four triangles, the common vertex being the one opposite to the selected edges.

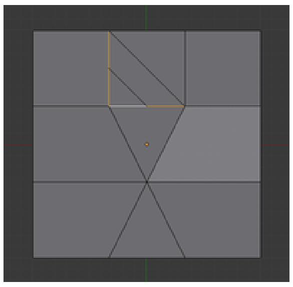
  Innervert
  : (i.e. “inner vertex”), The selected edges are sub-divided, then an edge is created between the two new vertices, creating a small triangle. This edge is also sub-divided, and the “inner vertex” thus created is linked by another edge to the one opposite to the original selected edges. All this results in a quad sub-divided in a triangle and two quad.

  Path
  : First an edge is created between the two opposite ends of the selected edges, dividing the quad in two triangles. Then, the same goes for the involved triangle as described above.

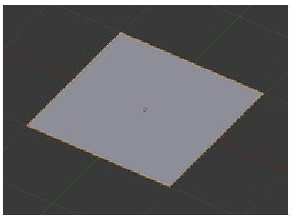
  Straight Cut
  : Currently non functioning...

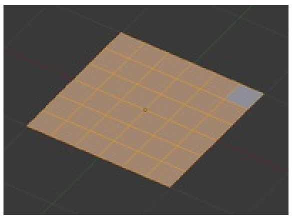

Fan cut type

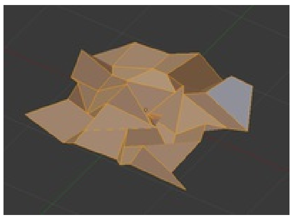

Innervert cut type


Path cut type

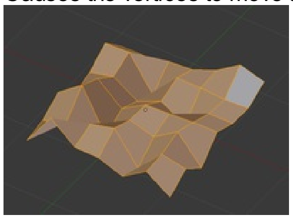
Fractal
: Displaces the vertices in random directions after the mesh is subdivided


Plane before subdivision

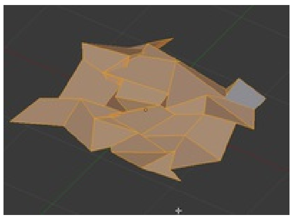

Regular subdivision


Same mesh with fractal added

Along Normal
: Causes the vertices to move along the their normals, instead of random directions

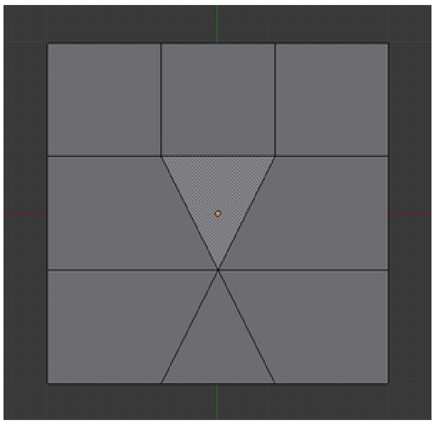

Along normal set to 1

Random Seed
: Changes the random seed of the noise function, producing a different result for each seed value.

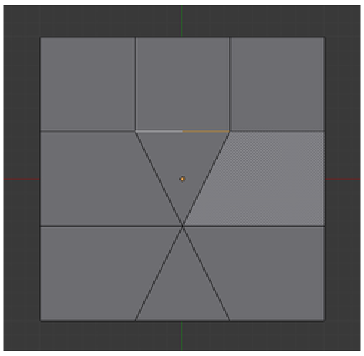

Same mesh with a different seed value

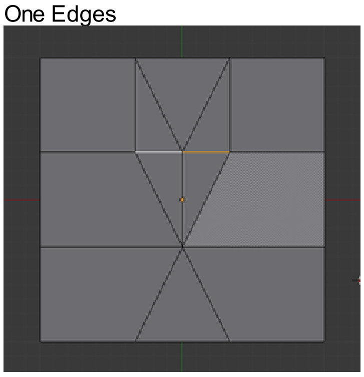
Examples

Below are several examples illustrating the various possibilities of the Subdivide and Subdivide Multi tools. Note the selection after subdivision.

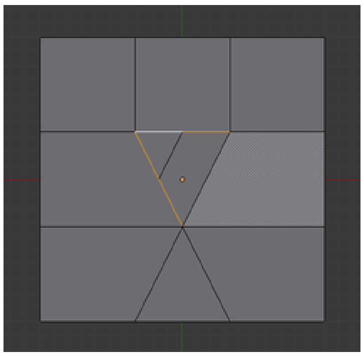

The sample mesh.

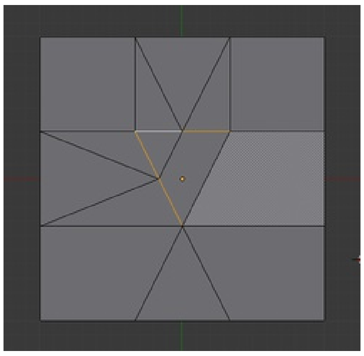
One Edge


One Edges

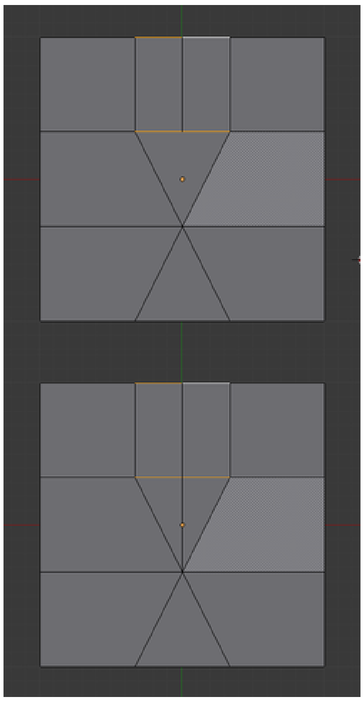

Quad/Tri Mode

Two Tri Edges

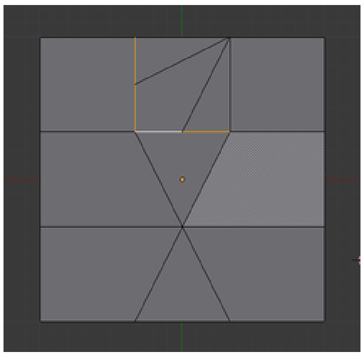


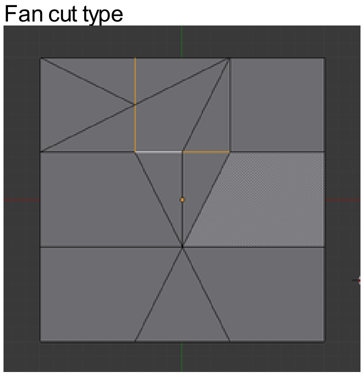

Quad/Tri Mode

Two Opposite Quad Edges

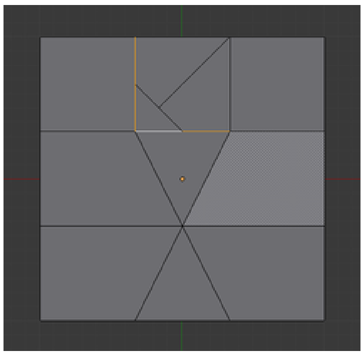
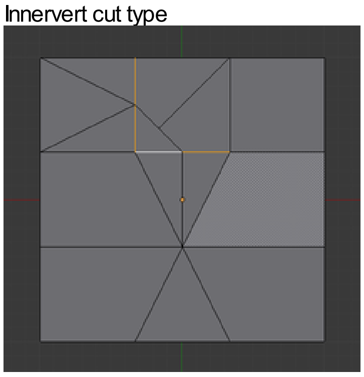

Quad/Tri Mode

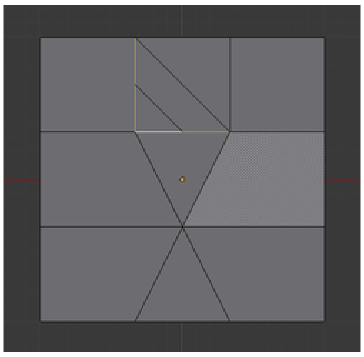
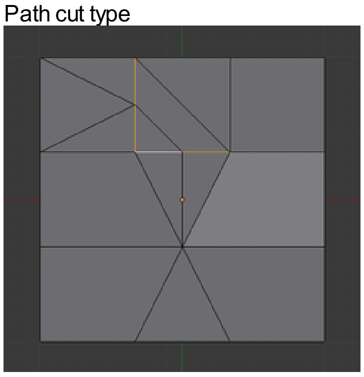
Two Adjacent Quad Edges


Fan cut type

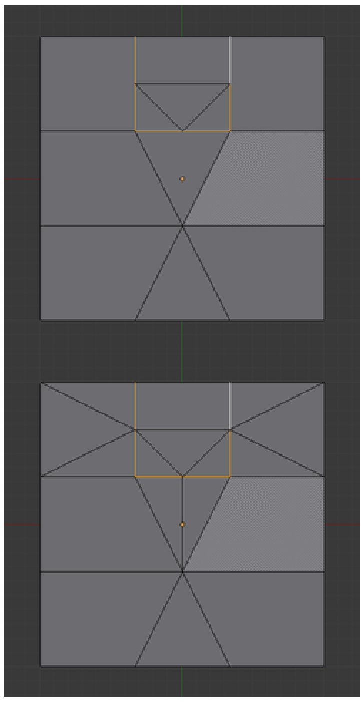


Quad/Tri Mode

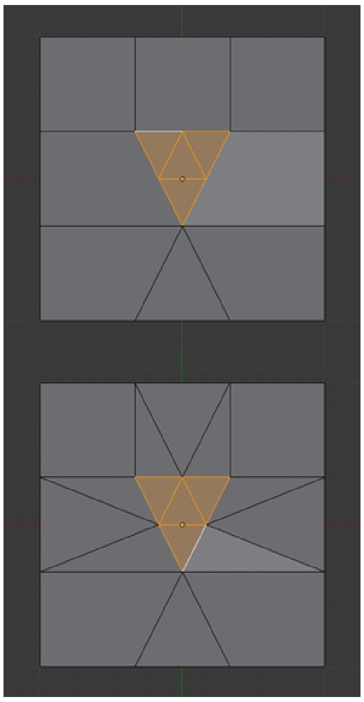

Innervert cut type

Quad/Tri Mode


Path cut type

Quad/Tri Mode

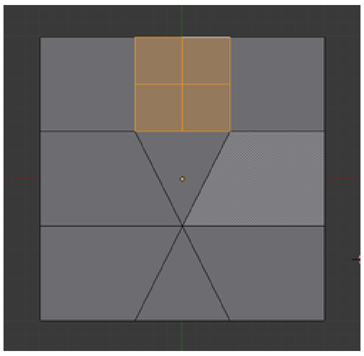
Three Edges

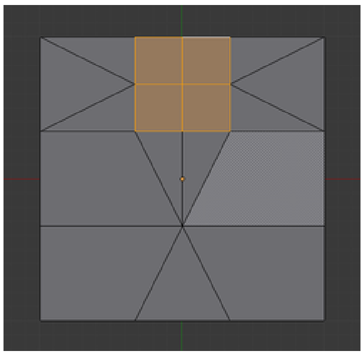

Quad/Tri Mode

Tri

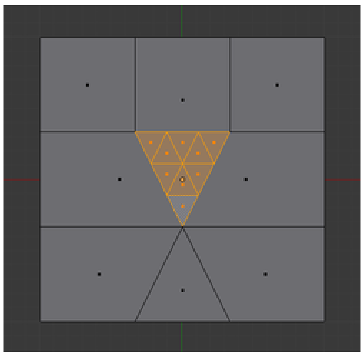

Quad/Tri Mode

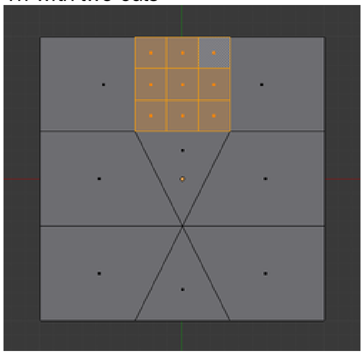
Quad/Four Edges




Quad/Tri Mode

Multicut


Tri with two cuts


Quad with two cuts

Loop Subdivide

Mode: Edit mode

Panel: Editing context → Mesh Tools

Hotkey: CtrlR

Loop Cut splits a loop of faces by inserting a new edge loop intersecting the chosen edge. The tool is interactive and has two steps:

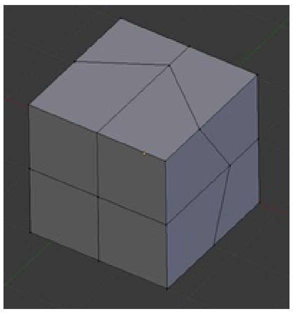
Usage

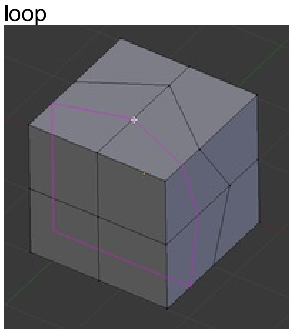
Pre-visualizing the cut

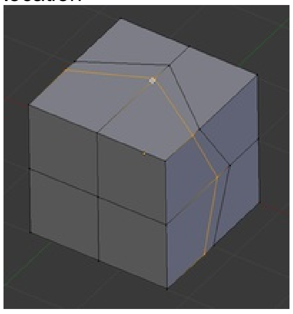
After the tool is activated, move the cursor over a desired edge. The cut to be made is marked with a magenta colored line as you move the mouse over the various edges. The to-be-created edge loop stops at the poles (tris and ngons) where the existing face loop terminates.

Sliding the new edge loop

Once an edge is chosen via LMB, you can move the mouse along the edge to determine where the new edge loop will be placed. This is identical to the Edge Slide tool. Clicking LMB again confirms and makes the cut at the pre-visualized location, or clicking RMB forces the cut to exactly 50%. This step is skipped when using multiple edge loops (see below)


mesh before inserting edge loop


Preview of edge loop location


Interactive placement of edge loop between adjacent loops

Options

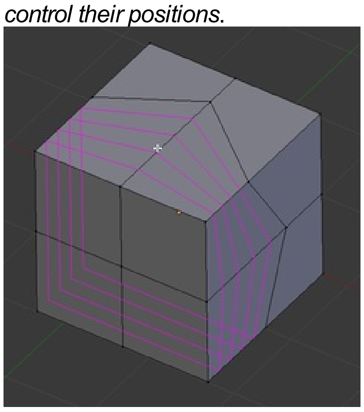
Options are only available while the tool is in use, and are displayed in the 3d view header

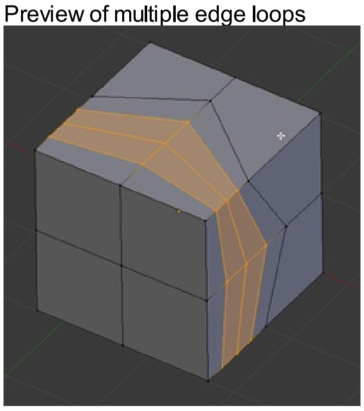
Even E
: Only available for single edge loops. This matches the shape of the edge loop to one of the adjacent edge loops. (See Edge Slide tool for details)

Flip F
: When Even is enabled, this flips the target edge loop to match. (See Edge Slide tool for details)

Number of Cuts Wheel or + NumPad/- NumPad

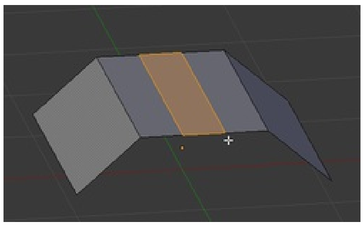
After activating the tool, but before confirming initial loop location, you can increase and decrease the number of cuts to create, by entering a number with the keyboard, scrolling Wheel or using + NumPad and - NumPad.
Note that when creating multiple loops, these cuts are uniformly distributed in the original face loop, and *you will not be able to control their positions*.

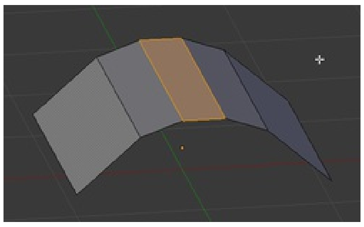

Preview of multiple edge loops


Result of using multiple cuts

Smoothing Alt Wheel

Smoothing causes edge loops to be placed in an interpolated position, relative to the face it is added to, causing them to be shifted outwards or inwards by a given percentage, similar to the Subdivide Smooth command. When not using smoothing, new vertices for the new edge loop are placed exactly on the pre-existing edges. This keeps subdivided faces flat, but can distort geometry, particularly when using Subdivision Surfaces. Smoothing can help maintain the curvature of a surface once it is subdivided.


Added edge loops without smoothing


Same edge loops, but with smoothing value

Knife Tool

Mode: Edit mode

Panel: Mesh Tools (Editing context, F9)

Hotkey: K or ⇧ ShiftK

The Knife Tool has been improved for Blender 2.6. It subdivides edges and faces intersected by a user-drawn “knife” line. The tool is now fully interactive, and snaps to edges, cut lines, and vertices, and can create multiple cuts on an edge.

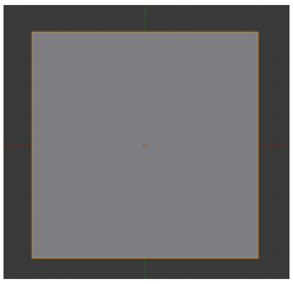
For example, if you wish to cut a hole in the front of a sphere, you select only the front edges, and then draw a line over the selected edges with the mouse. The tool is interactive, and works on primary edges, selected either implicitly by selecting all, or explicitly by box-selecting or ⇧ Shift RMB -clicking a few edges.

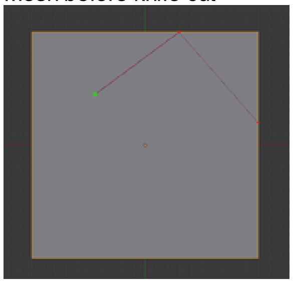
Use ⇧ ShiftK or the Select tool in the tool panel to force the knife tool to work only on a selection and in cut-through mode (see below).

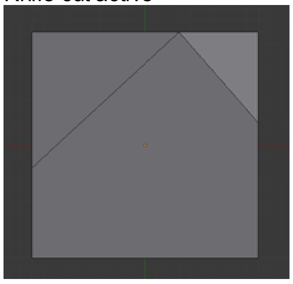
Usage

When you press K (or ⇧ ShiftK), the Knife tool becomes active.

Drawing the cut line
: When using Knife Subdivide, the cursor changes to an icon of a scalpel and the header changes to display options for the tool. You can draw connected straight lines by clicking LMB .

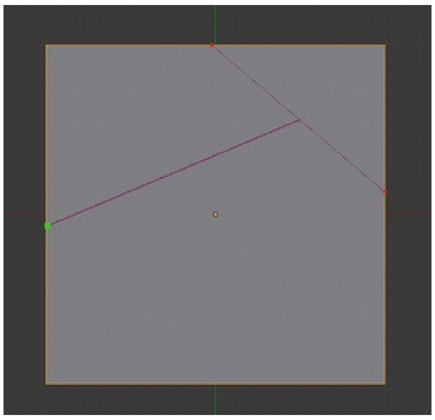

Mesh before knife cut

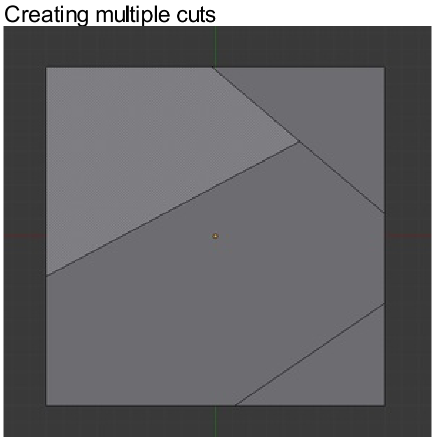

Knife cut active


After confirming knife cut

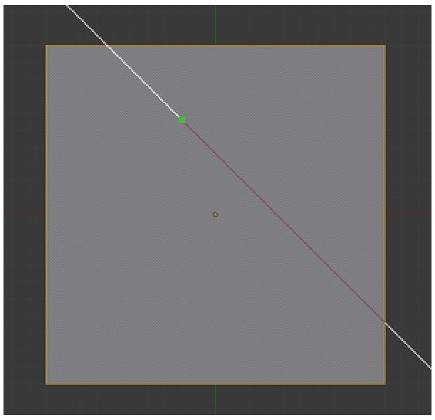
Options

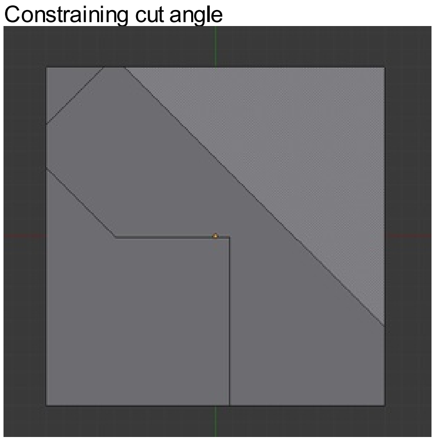
New cutE
: Begins a new cut. This allows you to define multiple distinct cut lines. If multiple cuts have been defined, they are recognized as new snapping points.


Creating multiple cuts


Result of starting new cuts while in the tool

Midpoint snap Ctrl
    Hold to snap the cursor to the midpoint of edges
Ignore snap⇧ Shift
    Hold to make the tool ignore snapping.
Angle constrainC
    Hold to constrain the cut vector to the view in 45 degree increments.


Constraining cut angle


Result of constraining cut angle

Cut throughZ
Allow the cut tool to cut through to obscured faces, instead of only the visible ones.

Confirming and selection

Pressing Esc or RMB at any time cancels the tool, and pressing ↵ Enter confirms the cut, with the following options:

↵ Enter will leave selected every edge except the new edges created from the cut.

Limitations

If you try to make cuts that end off in the middle of a face, those cuts are ignored. This is a limitation of the current geometry that can be modeled in Blender.

Closed cycles can be cut in the middle of a face, forming holes, but those holes will be connected to the surrounding geometry by two edges, for similar modeling limitation reasons.

In 'cut through' mode, only cut lines that completely cross faces will make cuts.

Optimizations

With a large mesh, it will be quicker to select a smaller number of vertices—those defining only the edges you plan to split—so that the Knife will save time in testing selected vertices for knife trail crossings.

Knife Project

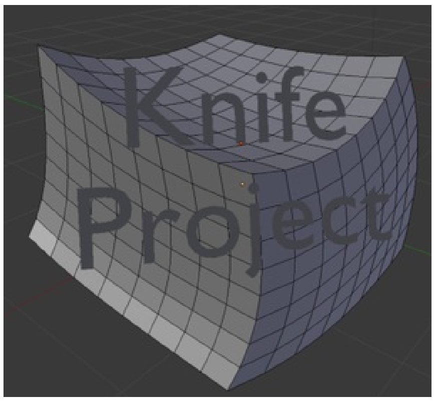
Knife projection is a non-interactive tool where you can use objects to cookie-cut into the mesh rather than hand drawing the line.

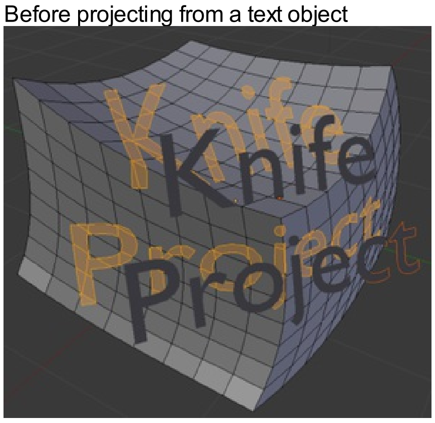
This works by using the outlines of other selected objects in edit-mode to cut into the mesh, resulting geometry inside the cutters outline will be selected.

Outlines can be wire or boundary edges.

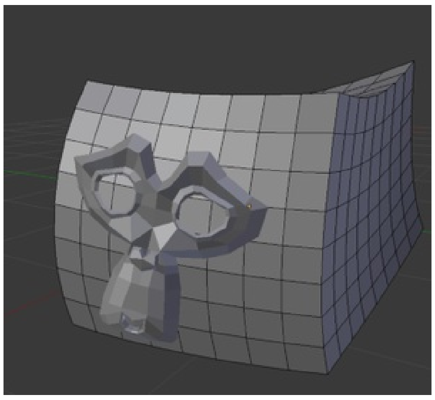
To use Knife Project, in 'object' mode select the "cutting object" first then shift select the "object to be cut". Now tab into edit mode and press "knife project".

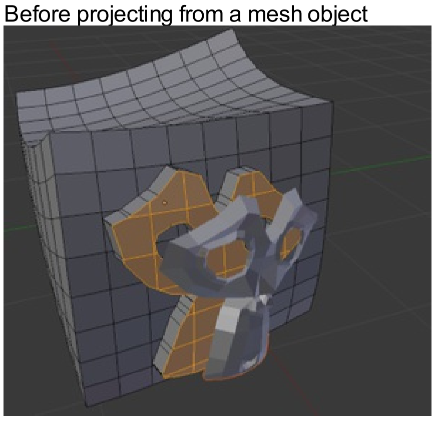
Examples

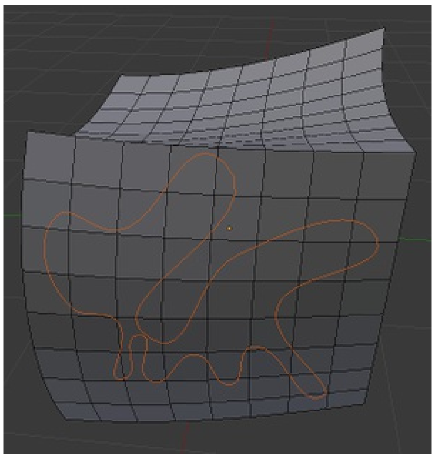


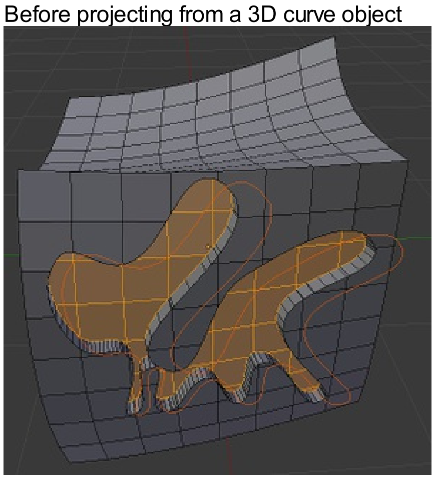
Before projecting from a text object


Resulting knife projection


Before projecting from a mesh object


Resulting knife projection (extruded after)


Before projecting from a 3D curve object


Resulting knife projection (extruded after)

Known Issues

Cutting holes into single faces may fail, this is the same limitation as with the regular knife tool but more noticeable for text, this can be avoided by projecting onto more highly subdivided geometry.

Mesh Bisect

Mode: Edit mode

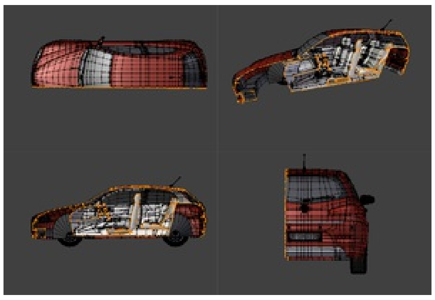
Menu: Mesh » Bisect

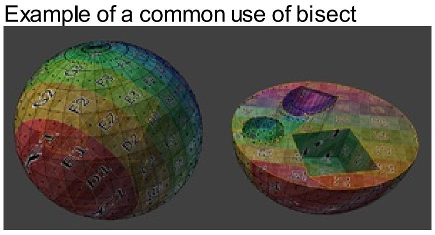
The bisect tool is a quick way to cut a mesh in-two along a custom plane.

There are three important differences between this and the knife tool.

- The plane can be numerically adjusted in the operator panel for precise values.
- Cuts may remove geometry on one side.
- Cuts can optionally fill in the holes created, with materials and UV's & vertex-colors based on the surrounding geometry.

This means the bisect tool can cut off parts of a mesh without creating any holes.


Example of a common use of bisect


Example of bisect with fill option

Vertex Connect

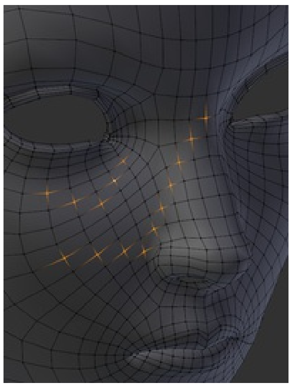
Mode: Edit mode

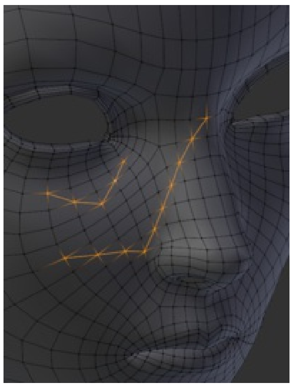
Hotkey: J

Menu: Mesh → Vertices → Connect

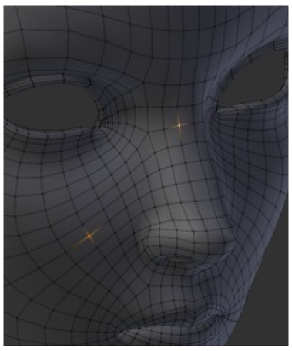
This tool joins selected vertices by edges, The main difference between this and creating edges is that faces are split by the newly joined vertices.

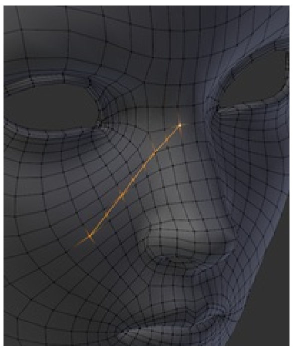
When many vertices are selected, faces will be split by their selected vertices.


Before


After

When there are only 2 vertices selected, a cut will be made across unselected faces, a little like the knife tool; however this is limited to straight cuts across connected faces.


Before

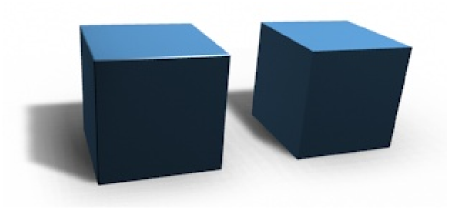

After

Bevel

Mode: Edit mode

Hotkey: CtrlB or W » Bevel

Menu: Mesh » Edges » Bevel or CtrlE » Bevel


With bevel and without bevel.

The bevel tool allows you to create chamfered or rounded corners to geometry. A bevel is an effect that smooths out edges and corners. True world edges are very seldom exactly sharp. Not even a knife blade edge can be considered perfectly sharp. Most edges are intentionally beveled for mechanical and practical reasons.

Bevels are also useful for giving realism to non-organic models. In the real world, the blunt edges on objects catch the light and change the shading around the edges. This gives a solid, realistic look, as opposed to un-beveled objects which can look too perfect.

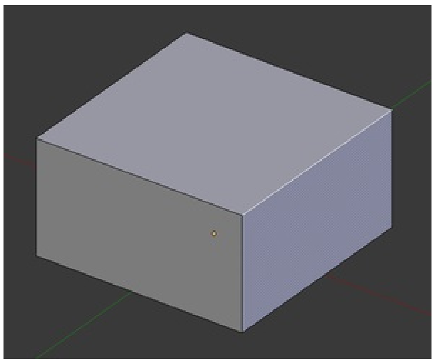
Bevel Modifier

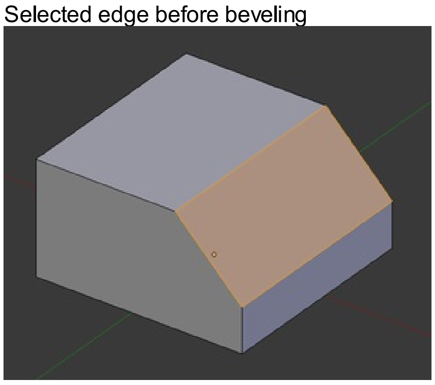
The Bevel Modifier is a non destructive alternative to the bevel tool. It gives you the same options, with additional goodies, like the bevel width controlled by the vertices weight, and all the modifiers general enhancements (non-destructive operations, …). Note that the Bevel modifier has no recursive option. To overcome this, you can add additional modifiers to multiply the effect.

Usage

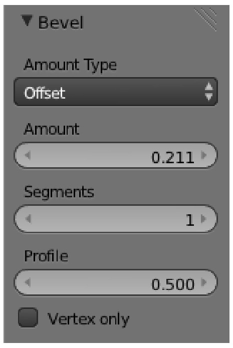
The Bevel tool works only on selected edges. It will recognize any edges included in a vertex or face selection as well, and perform the bevel the same as if those edges were explicitly selected. The Bevel tool smooths the edges and/or “corners” (vertices) by “subdividing” them a specified number of times (see the options below for details about the bevel algorithm).

Use CtrlB or a method listed above to run the tool. Move the mouse to interactively specify the bevel offset, and scroll the Wheel to increase or decrease the number of segments. (see below)


Selected edge before beveling


Result of bevel

Options


▼ Bevel

Amount Type

Offset

Amount

0.211

Segments

1

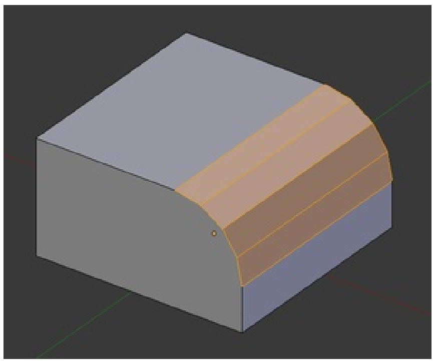
Profile

0.500

Vertex only

Offset

You can change the bevel width by moving the mouse towards and away from the object, a bit like with transform tools. As usual, the scaling can be controlled to a finer degree by holding ⇧ Shift to scale in 0.001 steps. LMB finalizes the operation, RMB or Esc aborts the action.

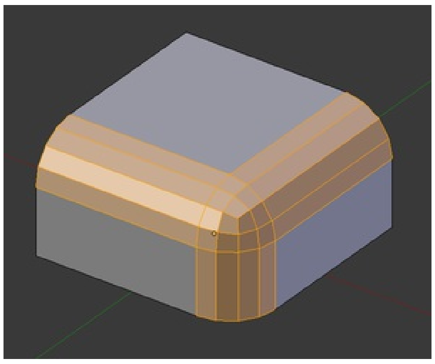

Bevel with 4 segments

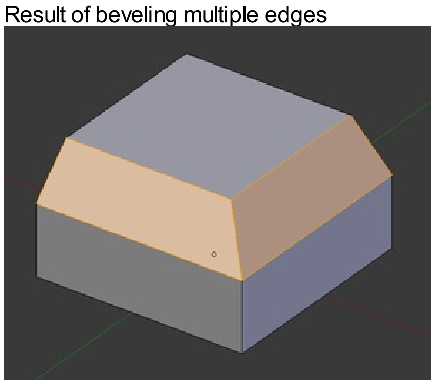
Segments

The number of segments in the bevel can be defined by scrolling the mouse Wheel to increase or decrease this value. The greater the number of recursions, the smoother the bevel.

Alternatively, you can manually enter a scaling value while using the tool, or in the Mesh Tool options panel after using the tool.

Examples


Result of beveling multiple edges


Another example of beveling multiple edges

Miscellaneous Editing Tools

Sort Elements

This tool (available from the Specials, Vertices, Edges and Faces menus) allows you to reorder the matching selected mesh elements, following various methods. Note that when called from the Specials menu, the affected element types are the same as the active select modes.

View Z Axis
: Sort along the active view's Z axis, from farthest to nearest by default (use Reverse if you want it the other way).

View X Axis
: Sort along the active view's X axis, from left to right by default (again, there's the Reverse option).

Cursor Distance
: Sort from nearest to farthest away from the 3D cursor position (Reverse also available).

Material
: **Faces only!** Sort faces from those having the lowest material's index to those having the highest. Order of faces inside each of those "material groups" remains unchanged. Note that the Reverse option only reverses the order of the materials, *not* the order of the faces inside them.

Selected
: Move all selected elements to the beginning (or end, if Reverse enabled), without affecting their relative orders. **Warning: this option will also affect unselected elements' indices!**

Randomize
: Randomizes indices of selected elements (*without* affecting those of unselected ones). The seed option allows you to get another randomization – the same seed over the same mesh/set of selected elements will always give the same result!

Reverse
: Simply reverse the order of the selected elements.

Retopologizing

Note

In Blender 2.5, the Retopo tool has been replaced by improved mesh snapping functionality. This page will change as retopology tools are updated in newer versions of Blender

Retopology is a common part of modeling workflows. Often times, a model is created with emphasis on form and detail, however, its topology, or edge flow is not ideal, or the mesh is very dense, and not efficient. Modelers can create a new lower resolution mesh that matches the form of the original mesh.

Mesh Snapping

By enabling snapping, and setting the snap element to Face, mesh vertices will be projected onto the closest surface in the viewport, in the view's Z-axis.

This allows you to model freely, without concern for form, and to focus on topology

See Snapping

Shrinkwrap Modifier

The Shrinkwrap Modifier is useful in conjunction with face snapping. If edits to the new mesh have been made with snapping disabled, the shrinkwrap modifier will allow you to stick the new mesh to the old mesh, as if you were shrinkwrapping it.

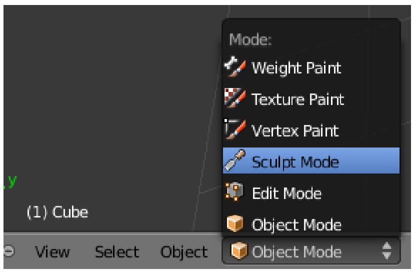
Overview

Sculpt Mode is similar to Edit Mode in that it is used to alter the shape of a model, but Sculpt Mode uses a very different workflow: instead of dealing with individual elements (vertices, edges, and faces), an area of the model is altered using a brush. In other words, instead of selecting a group of vertices, Sculpt Mode automatically selects vertices based on where the brush is, and modifies them accordingly.

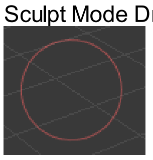
Sculpt Mode

Sculpt mode is selected from the mode menu of the 3D View header.

Once sculpt mode is activated the Toolbar of the 3D View will change to sculpt mode specific panels. The panels in the toolbar will be Brush, Texture, Tool, Symmetry, Stroke, Curve, Appearance, and Options. Also a red circle will appear that follows the location of the cursor in the 3d view.



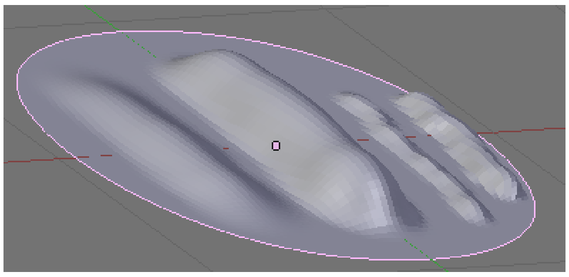
Sculpt Mode Dropdown.


The cursor in Sculpt Mode.

Sculpt Brushes

Brushes are brush presets. They are a combination of a 'tool', along with stroke, texture, and options.

Sculpt Mode has sixteen brushes, each of which operates on the model in a unique way. Many can be toggled to have an additive or subtractive effect. They can be selected in the Tool menu. The current Brush presets are: Blob, Clay, Crease, Draw, Fill/Deepen, Flatten/Contrast, Grab, Inflate/Deflate, Layer, Nudge, Pinch/Magnify, Rotate, Scrape/Peak, Smooth, Snake Hook, Thumb:


Drawing in various sizes and strengths.

Blob
: Pushes mesh outward or inward into a spherical shape with settings to control the amount of pinching at the edge of the sphere.

Clay (C)
: Similar to the Draw brush, but includes settings to adjust the plane on which the brush acts.

Clay Strips
: Similar to the Clay brush, but it uses a cube test to define the brush area of influence rather than a sphere.

Crease
: Creates sharp indents or ridges by pushing or pulling the mesh, while pinching the vertices together.

Draw (D)
: Moves vertices inward or outward, based the average normal of the vertices contained within the drawn brush stroke.

Fill
: The Fill brush works like the Flatten brush, but only brings vertices below the brush plane upwards. The inverse of the Scrape brush is to Deepen by pushing vertices above the plane downward.

Flatten (T)
: The Flatten brush finds an 'area plane' located by default at the average height above/below the vertices within the brush area. The vertices are then pulled towards this plane. The inverse of the Flatten brush is the Contrast brush which pushes vertices up

or down away from the brush plane.

Grab (G)
: Grab is used to drag a group of points around. Unlike the other brushes, Grab does not modify different points as the brush is dragged across the model. Instead, Grab selects a group of vertices on mousedown, and pulls them to follow the mouse. The effect is similar to moving a group of vertices in Edit mode with proportional-editing enabled, except that Grab can make use of other Sculpt Mode options (like textures and symmetry).

Inflate (I)
: Similar to Draw, except that vertices in Inflate mode are displaced in the direction of their own normals.

Layer (L)
: This brush is similar to Draw, except that the height of the displacement layer is capped. This creates the appearance of a solid layer being drawn. This brush does not draw on top of itself; a brush stroke intersects itself. Releasing the mouse button and starting a new stroke will reset the depth and paint on top of the previous stroke.

Nudge
: Moves vertices in the direction of the brush stroke.

Pinch (P)
: Pinch pulls vertices towards the center of the brush. The inverse setting is Magnify, in which vertices are pushed away from the center of the brush.

Rotate
: Rotates vertices within the brush in the direction the cursor is moved.

Scrape
: The Scrape brush works like the Flatten brush, but only brings vertices above the plane downwards. The inverse of the Scrape brush is to Peak by pushing vertices above the plane up away from the plane.

Smooth (S)
: As the name suggests, eliminates irregularities in the area of the mesh within the brush’s influence by smoothing the positions of the vertices.

Snake Hook
: Pulls vertices along with the movement of the brush to create long, snake-like forms.

Thumb
: Similar to the Nudge brush, this one flattens the mesh in the brush area, while moving it in the direction of the brush stroke.

Sculpting with the Multires Modifier

...

Sculpt Properties Panel

This panel appears in the tool palette on the left side of the 3D viewport.

Brush Menu

Radius
: This option controls the radius of the brush, measured in pixels. F in the 3D view allows you to change the brush size interactively by dragging the mouse and then left clicking (the texture of the brush should be visible inside the circle). Typing a number then enter while in F sizing allows you to enter the size numerically. Brush size can be affected by enabling the pressure sensitivity icon, if a supported tablet is being used.

Strength
: Strength controls how much each application of the brush affects the model. For example, higher values cause the Draw brush to add depth to the model more quickly, and cause the Smooth brush to smooth the model more quickly. This setting is not available for Grab, Snake Hook, or Rotate.

If the range of strengths doesn’t seem to fit the model (for example, if even the lowest strength setting still makes too large of a change on the model) then you can scale the model (in Edit Mode, not Object Mode). Larger sizes will make the brush’s effect smaller, and vice versa. You can change the brush strength interactively by pressing ⇧ ShiftF in the 3D view and then moving the brush and then left clicking. You can enter the size numerically also while in ⇧ ShiftF sizing. Brush strength can be affected by enabling the pressure sensitivity icon, if a supported tablet is being used.

Autosmooth
: Sets the amount of smoothing to be applied to each stroke.

Sculpt Plane
: Use this menu to set the plane in which the sculpting takes place.

Plane Offset
: Adjusts the plane on which the brush acts toward or away from the viewer.

Trim
Enables trimming of the sculpt plane, determined by the Distance setting.

Front Faces Only
When enabled, the brush only affects vertices that are facing the viewer.

Accumulate
Causes stroke dabs to accumulate on top of each other.

Stroke Menu

Stroke Method
Defines the way brush strokes are applied to the mesh:

Dots
Standard brush stroke.

Drag Dot
Creates a single displacement in the brush shape. Click then drag on mesh to desired location, then release.

Space
Creates brush stroke as a series of dots, whose spacing is determined by the Spacing setting. Spacing represents the percentage of the brush diameter.

Anchored
Creates a single displacement at the brush location. Clicking and dragging will resize the brush diameter. When Edge to Edge the brush location and orientation is determined by a two point circle, where the first click is one point, and dragging places the second point, opposite from the first.

Airbrush
Flow of the brush continues as long as the mouse click is held, determined by the Rate setting. If disabled, the brush only modifies the model when the brush changes its location. This option is not available for the Grab brush.

The following parameters are available for the Dots, Space, and Airbrushstrokes:

Smooth stroke
Brush lags behind mouse and follows a smoother path. When enabled, the following become active:

Radius
Sets the minimum distance from the last point before stroke continues.
Factor
Sets the amount of smoothing

Jitter
Jitters the position of the brush while painting.

Curve Menu

The Curve section allows you to use a curve control to the right to modify the intensity of the brush from its centre (left part of the curve) towards its borders (right part of the curve).

Texture Menu

A texture can be used to determine the strength of brush effects as well. Select an existing texture from the texture box, or create a new one by selecting the New button

Brush Mapping
Sets the way the texture is mapped to the brush stroke:

Fixed
If Fixed is enabled, the texture follows the mouse, so it appears that the texture is being dragged across the model.
Tiled
The Tile option tiles the texture across the screen, so moving the brush appears to move separately from the texture. The Tile option is most useful with tileable images, rather than procedural textures.
3D
The 3D option allows the brush to take full advantage of procedural textures. This mode uses vertex coordinates rather than the brush location to determine what area of the texture to use.

Angle
This is the rotation angle of the texture brush. It can be changed interactively via CtrlF in the 3D view. While in the interactive rotation you can enter a value numerically as well. Can be set to:

User
Directly input the angle value.

Rake
Angle follows the direction of the brush stroke. Not available with 3D textures.
Random
Angle is randomized.

Offset
Fine tunes the texture map placement in the x, y, and z axes.
Size
This setting allows you to modify the scaling factor of the texture. Not available for Drag textures.
Sample Bias
Value added to texture samples.
Overlay
When enabled, the texture is shown in the viewport, as determined by the ;**Alpha** value.

Symmetry Menu

Mirror the brush strokes across the selected local axes. Note that if you want to alter the directions the axes point in, you must rotate the model in Edit Mode, not Object Mode.

Feather
Reduces the strength of the stroke where it overlaps the planes of symmetry.
Radial
These settings allow for radial symmetry in the desired axes. The number determines how many times the stroke will be repeated within 360 degrees around the central axes.

Options Menu

Threaded Sculpt
Takes advantage of multiple CPU processors to improve sculpting performance.
Fast Navigation
For ;Multires models, show low resolution while navigation the viewport.
Show Brush

Shows the brush shape in the viewport.

Unified Settings

;**Size**

Forces the brush size to be shared across brushes.

;**Strength**

Forces the brush strength to be shared across brushes.

Lock
These three buttons allow you to block any modification/deformation of your model along selected local axes, while you are sculpting it.

Appearance Menu

You can set the color of the brush depending on if it is in additive or subtractive mode.

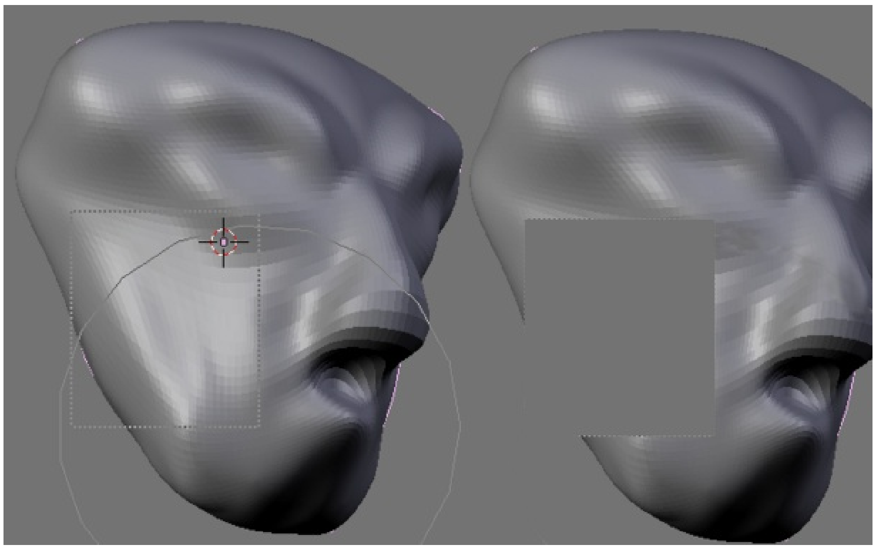
You can also set the brush icon from an image file.

Tool Menu

Here you can select the type of brush preset to use. Reset Brush will return the settings of a brush to its defaults. You can also set Blender to use the current brush for Vertex Paint mode, Weight Paint mode, and Texture Paint mode using the toggle buttons.

Hiding and Revealing Mesh

It is sometimes useful to isolate parts of a mesh to sculpt on. To hide a part of a mesh, press H then click & drag around the part you want to hide. To reveal a hidden part of a mesh, press ⇧ ShiftH then click & drag around the part you want to reveal. To reveal all hidden parts, just hit AltH.


Before and after Hiding.

Keyboard Shortcuts

These shortcuts may be customized under File > User preferences > Input > 3D View > Sculpt Mode.

Action	Shortcut
Hide mesh inside selection	H then click & drag
Reveal mesh inside selection	⇧ ShiftH then click & drag
Show entire mesh	AltH
Interactively set brush size	F
Increase/decrease brush size	[and]
Interactively set brush strength	⇧ ShiftF
Interactively rotate brush texture	CtrlF
Brush direction toggle (Add/Sub)	Ctrl pressed while sculpting
Set stroke method (airbrush, anchored, ..)	A
Smooth stroke toggle	⇧ ShiftS
Draw brush	D
Smooth brush	S
Pinch/Magnify brush	P
Inflate/Deflate brush	I
Grab brush	G
Layer brush	L
Flatten/Contrast brush	⇧ ShiftT
Clay brush	C
Crease brush	⇧ ShiftC
Snake Hook brush	K
Mask brush	M
Mask clear	AltM
Mask invert	CtrlI
Set brush by number	0 - 9 and ⇧ Shift0 to ⇧ Shift9
Sculpt options panel toggle	T
Step up one multires level	Page up
Step down one multires level	Page down
Set multires level	Ctrl0 to Ctrl5
Dynamic topology toggle	CtrlD
Set texture angle type	R
Translate/scale/rotate stencil texture	RMB, ⇧ Shift RMB, Ctrl RMB
Translate/scale/rotate stencil mask	Alt RMB, Alt⇧ Shift RMB, AltCtrl RMB

Page status (reviewing guidelines)

Page reviewed and in good shape

Removed from Blender 2.5
This feature is no longer available in Blender 2.5, see the Multires modifier.

Vertex Groups

- Create&Delete vertex groups
  General overview about creating and maintaining Vertex Groups
- Weight Edit
  editing Vetex group weights
- Weight Paint
  The weight Paint Mode
- Weight Paint Tools
  the Weight Paint tools

Vertex Groups

Vertex Groups are sets containing zero or more vertices of a Mesh Object or Lattice. A vertex can belong to zero or more Vertex Groups. The Vertex Groups could be used to tag vertices as belonging to a certain part of the mesh, think of the legs or seat of a chair, the whole chair, the hinges of a door, etc. Often they are used as a bridge between the vertices and deform bones; with a one on one relationship between the bones and Vertex groups, each vertex then belongs to all vertex groups for which it has a (non-zero) Weight Value (for example there could be a vertex group for the upper arm, and one for the lower arm, while vertices around the elbow would belong to both groups). Vertex Groups are therefore sometimes also named Weight Groups.

Vertex Groups are most commonly used for Armatures (See Skinning Mesh Objects). But they are also used in many other areas of Blender, like for example:

- Shape keys
- Modifiers
- Particle Generators
- Physics simulations

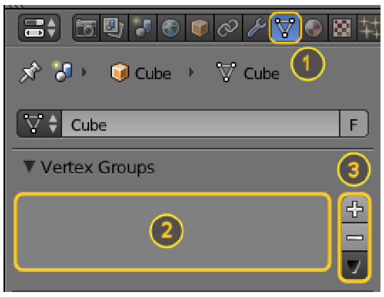
Many more usage scenarios are possible. Actually you can use Vertex Groups for whatever makes sense to you. In some contexts Vertex Groups can also be automatically generated (i.e. for rigged objects). However in this section we will focus on manually created (user-defined) Vertex Groups.

Vertex groups only apply to Mesh and Lattice Objects
Any other Object type has no vertices, hence it can not have Vertex Groups.

Typical usage scenarios for Vertex groups

- **Skinning an armature**
  If you want to animate your mesh and make it move, you will define an armature which consists of a bunch of bones. Vertex Groups are used to associate parts of the Mesh to Bones of the Armature, where you can specify an influence weight in the range [0.0 ... 1.0] for each vertex in the Vertex Group.
- **Working with Modifiers**
  Many modifiers contain the ability to control the modifier influence on each vertex separately. This is also done via Vertex Groups and the weight values associated to the vertices.
- **Quickly select/edit/hide parts of a mesh**
  By defining mesh regions with Vertex Groups you can easily select entire parts of your mesh with 3 clicks and work on them in isolation without having to create separate objects. With the hide function you can even remove a vertex group from the view (for later unhide).
- **Cull out and duplicate parts of a mesh**
  Consider modeling a Lego block. The most simple brick consists of a base and a stud (the bump to connect the bricks together). To create a four-stud block, you would want to be able to easily select the stud vertices, and, still in Edit mode, duplicate them and position them where you want them.

Creating Vertex Groups



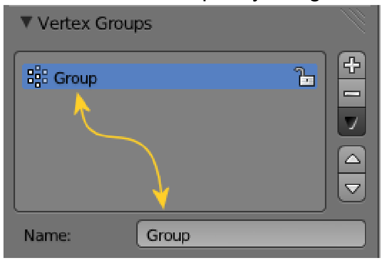
Empty Vertex Group Panel

Vertex Groups are maintained within the Object Data Properties window(1), and there in the Vertex Groups panel. As long as no Vertex groups are defined (the default for new Mesh Objects), the Panel is empty (2).

You create a vertex group by LMB the + button at the right Panel border (3). Initially the group is named `Group` (or `Group.nnn` when the name already exists) and gets displayed in the Panel (2) (see next image).

New Groups are always empty

You have to explicitly assign verts to the group as shown below.



Vertex Group Panel with one Group in Object mode

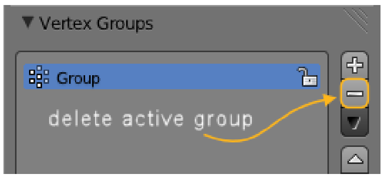
Group Name

As soon as the first Vertex Group is created a new input field shows up right below the panel. There you can change the Group name of the Group to your convenience.

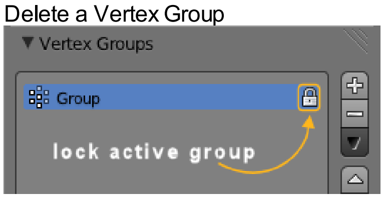
Active Group

When a Vertex Group is created, then it is also automatically marked as the Active Group. This is indicated by setting the background of the panel entry to a light blue color. If you have two or more groups in the list, then you can change the active group by LMB on the corresponding entry in the Vertex Group panel.

Deleting vertex Groups



Delete a Vertex Group



Lock a Vertex Group

You delete a Vertex Group by first making it the active group (select it in the panel) and then LMB the - button at the right Panel border.

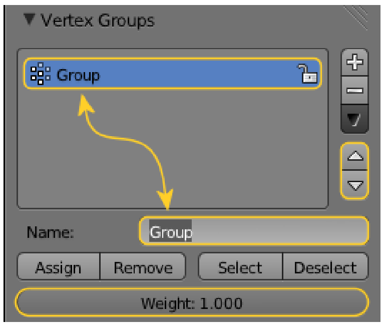
Deleting a Vertex Group only deletes the vertex assignments to the Group. The vertices themselves are not deleted.

Locking Vertex Groups

Right after creation of a Vertex Group, an open lock icon shows up on the right side of the Vertex Group List entry. This icon indicates that the Vertex Group can be edited. You can add vertex assignments to the group or remove assignments from the group. And you can change it with the weight paint brushes, etc.

When you click on the icon, it changes to a closed lock icon and all vertex group modifications get disabled. You can only rename or delete the group, and unlock it again. No other operations are allowed on locked Vertex Groups, thus all corresponding function buttons become disabled for locked Vertex Groups.

Working with Content of Vertex Groups



Vertex Group Panel in Edit Mode

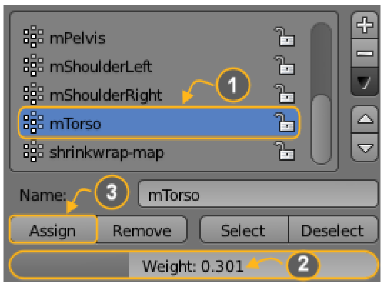
When you switch either to Edit-Mode or to Weight-Paint Vertex Selection Mode, then the Vertex Group panel expands and displays 2 more rows:

The first row contains 4 buttons for maintaining the Assign- and Select- status of vertices of the active Vertex Group:

- **Assign**: To assign the Selected vertices to the Group with the weight as defined in the "Weight:" input field (see below)
- **Remove**: To Remove the selected vertices from the Group (and thus also delete their weight values)
- **Select**: To Select all vertices contained in the Group
- **Deselect**: To deselect all verts contained in the group

Below this row of buttons you see a numeric "Weight:" input field where you specify the weight value that gets assigned to the selected verts when you press the Assign Button.

Assigning verts to a Group



Assign weights to active group

You add vertices to a group as follows:

1. Select the group from the group list, thus make it the Active Group (1).
2. From the 3D Viewport select ⇧ Shift RMB all vertices that you want to add to the group.
3. Set the weight value that shall be assigned to all selected verts (2).
4. LMB the Assign button to assign the selected verts to the active group using the given weight (3).

Note that weight Assignment is not available for locked Vertex Groups. The Assign button is grayed out in that case.

Assign is additive

The Assign button only adds the currently selected vertices to the active group. Vertices already assigned to the group are not removed from the group. Also keep in mind that a vertex can be assigned to multiple groups.

Checking assignments

To be sure the selected verts really have been added to the Vertex Group, you can try the deselect button. If the verts do not get deselected, then you probably forgot to hit the Assign button. But you can do that safely now. But remind: All selected verts get the weight assigned as displayed in the "Weight:" input field!

Removing assignments from a Group

You remove vertices from a group as follows:

1. Select the group from the group list (make it the active group).
2. ⇧ Shift RMB all vertices that you want to remove from the group.
3. LMB click the Remove button.

Note that Removing weight Assignments is not available for locked Vertex Groups. The Remove button is grayed out in that case.

Using groups for Selecting/Deselecting

You can quickly select all assigned vertices of a group:

1. (optionally) press A once or twice to unselect all vertices.
2. Select the group from the group list (make it the active group).
3. When you now LMB click the Select button, then the vertices assigned to the active group will be selected and highlighted in the 3D Viewport.
4. When you LMB click the Deselect button instead, then the vertices assigned to the active group will be deselected in the 3D Viewport.

Selecting/Deselecting is additive

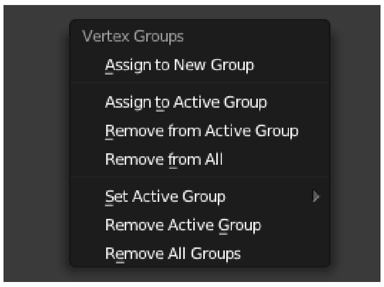
If you already have verts selected in the 3D View, then selecting the verts of a group will add the verts but also keep the already-selected verts selected. Vice versa, deselecting the verts of a vertex group will only deselect the verts assigned to the group and keep all other verts selected.

Finding ungrouped verts

You can find ungrouped vertices as follows:

1. Press A once or twice to unselect all vertices.
2. In the footer of the 3D Viewport: Navigate to Select -> Ungrouped verts

Keyboard Shortcuts



Vertex Groups popup menu

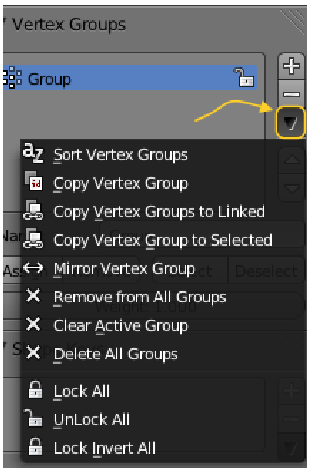
In Edit Mode you can type CtrlG to a shortcut Menu for adding/removing verts to/from groups. The popup menu provides the following functions with obvious functionality:

- Assign to New Group
- Assign to Active Group
- Remove from Active Group
- Remove from All

The following functions should not be located here and might be removed in a future version of Blender:

- Set Active Group
- Set Remove Acive Group
- Set Remove All Groups

Vertex Group Management



Vertex groups panel's dropdown menu

Vertex Groups provide a more complex set of functions inside a Pull down menu. This menu is accessible from the Vertex Group Panel by clicking on the dark gray arrow down icon on the right panel border.

The following functions of the Pulldown Menu operate on the assigned vertices:

Sort Vertex Groups
: Sorts Vertex Groups Alphabetically

Copy Vertex Group
: Add a Copy of the active Vertex Group as a new Group. The new group will be named like the original group with "_copy" appended at the end of its name. And it will contain associations to exactly the same verts with the exact same weights as in the source vertex group.

Copy Vertex Groups to Linked
: Copy Vertex Groups of this Mesh to all linked Objects which use the same mesh data (all users of the data).

Copy Vertex Group to Selected
: Copy all Vertex Groups to other Selected Objects provided they have matching indices (typically this is true for copies of the mesh which are only deformed and not otherwise edited).

Mirror Vertex Group
: Mirror all Vertex Groups, flip weights and/or names, editing only selected vertices, flipping when both sides are selected; otherwise copy from unselected. Note: this function will be reworked (and fully documented) in a future release.

Remove from All Groups (not available for locked groups)
: Unassign the selected Vertices from all groups. After this operation has been performed, the verts will no longer be contained in any vertex group.

Clear Active group (not available for locked groups)
: Remove all assigned vertices from the active Group. The group is made empty. Note that the vertices may still be assigned to other Vertex Groups of the Object.

Delete All Groups
: Remove all Vertex Groups from the Object.

The following functions operate only on the lock state settings:

Lock All
: Lock all groups

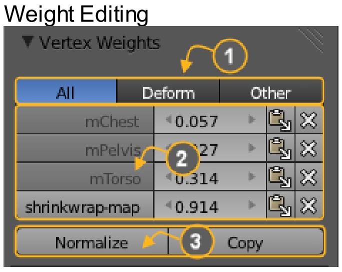
Unlock All
: Unlock all groups

Lock_Invert All
: Invert Group Locks

Hints

- Multiple objects sharing the same mesh data have the peculiar property that the group names are stored on the object, but the weights in the mesh. This allows you to name groups differently on each object, but take care because removing a vertex group will remove the group from all objects sharing this mesh.

Weight Editing



Vertex Weights Panel

As mentioned before in Vertex Groups each entry in a Vertex Group also contains a weight value in the range of [0.0,1.0]. Blender provides a Vertex Weights panel from where you can get (and edit) information about the weight values of each Vertex of a mesh. That is: to which Vertex Groups the vertex is assigned with which weight value.

The Vertex Weights panel can be found in the left property sidebar of the 3D Viewport. It is available in Edit mode and in Weight Paint mode (when Vertex Selection masking is enabled as well). The panel is separated into the sections

- Vertex Group Categories (1)
- Weight Table (2)
- function bar (3)

Vertex Group Categories

Well, actually we do not have any strict categories of Vertex Groups in Blender. Technically they all behave the same way. However we can identify 2 implicit categories of Vertex Groups:

The Deform Groups

These Vertex groups are sometimes also named Weight Groups. They are used for defining the weight tables of Armature bones. All Deform Groups of an Object are strictly related to each other via their weight values.

Strictly speaking, the sum of all deform weights for any vertex of a mesh should be exactly 1.0. In Blender this constraint is a bit relaxed (see below). Nevertheless, Deform Groups should always be seen as related to each other. Hence we have provided a filter that allows restricting the Vertex Weight panel to display only the Deform bones of an Object.

The Other Groups

All other usages of Vertex Groups are summarized into the Other category. These vertex groups can be found within Shape keys, Modifiers, etc... There is really no good name for this category, so we kept it simple and named it Other.

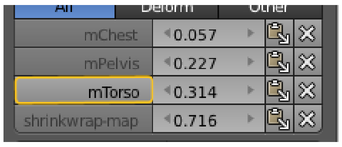
The Weight Table

The Weight Table shows all weights associated to the active vertex. Note that a vertex does not necessarily have to be associated to any vertex groups. In that case the Vertex Weights Panel is not displayed.

The active Vertex

That is the most recently selected vertex. This vertex is always highlighted so that you can see it easily in the mesh. If the active Vertex does not have weights, or there is no active vertex selected at the moment, then the Vertex Weights Panel disappears.

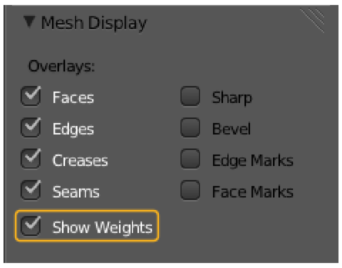
Each row in the Weight table contains 4 active elements:



Change Active Group

Set the Active Group

As soon as you select any of the Vertex Group Names in the Weight table, the referenced Vertex Group becomes the new Active group.

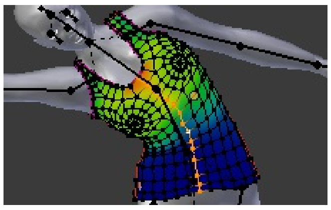


Enable display of Weights in Edit Mode

Display Weights in Edit Mode

When you are in edit mode, you can make the Weights of the active Group visible on the mesh:

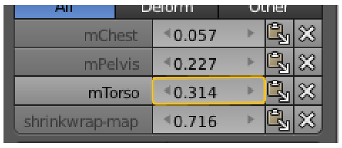
Search the Mesh Display panel in the Properties sidebar. And there enable the Show Weights option. Now you can see the weights of the active Vertex Group displayed on the mesh surface.


Weights in Edit Mode

Edit Weights in Edit Mode

It is now very easy to work with weightmaps in Edit mode. All edit options of the mesh are available and you have direct visual control over how your Weights change when you edit the weight values.

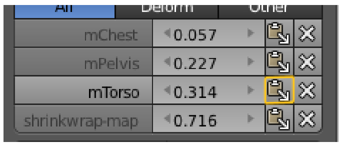
Note that for the deformation to be visible in edit mode, the *Use modifier while in Edit mode* and the *Apply modifier to editing cage during Edit mode* icons of the Armature modifier (show modifiers with the wrench icon in the Properties Panel on the left) must be selected.



Change Weight Value

Change a weight

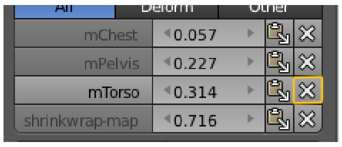
You can either enter a new weight value manually (click on the number and edit the value), or you can change the weight by LMB and while holding down the mouse button, drag right or left to increase/decrease the weight value. You also can use the right/left arrows displayed around the weight value to change the weight in steps.



Paste weights

Paste a weight to other verts

LMB the Paste Icon allows you to forward a single weight of the active Vertex to all selected vertices. But note that weights are only pasted to verts which already have a weight value in the affected Vertex Group.

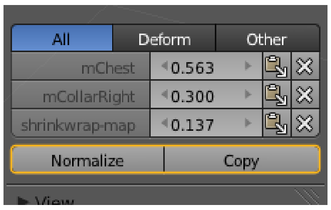


Delete weights

Delete a weight from a Group

LMB the Delete Icon will instantly remove the weight from the active vertex. thus the entire row disappears when you click on the delete icon.

The Function bar



Vertex Weights panel Function Bar

The function bar contains 2 functions:

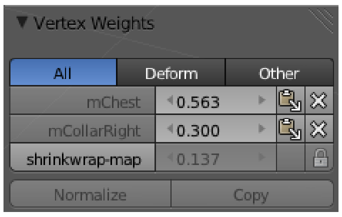
Normalize
: Normalizes the weights of the active Vertex. That is all weights of the active vertex are recalculated such that their relative weight is maintained and the weight sum is 1.0.

Copy
: Copies all weights defined for the active Vertex to all selected Verts. Thus all previously defined weights are overwritten.

The filter setting is respected

Note that both functions only work on the Vertex Groups currently displayed in the Weights Table. So if for example only the Deform weights are displayed, then Normalize and Copy only affect the Deform bones.

About locked Vertex Groups



Vertex Weights panel Locked

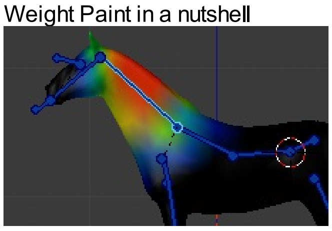
Whenever a Weight Group is locked, all data changing functions get disabled:

- Normalize the vertex Weights.
- Copy the Vertex weights.
- Change the Weight of the active vert.
- Paste to selected verts.

The filter setting is respected

If you have for example all deform weight groups unlocked and all other vertex groups locked, then you can safely select Deform from the Filter row and use all available functions from the Weight table again.

Weight Paint in a nutshell



- You enter Weight Paint mode from the Mode Menu (hotkey=Ctrl⇆ Tab). The selected Mesh Object is displayed slightly shaded with a rainbow color spectrum.
- The color visualizes the weights associated to each vertex in the active Vertex Group. Blue means unweighted; Red means fully weighted.
- You can customize the colors in the weight gradient by enabling Custom Weight Paint Range in the System tab of the User Preferences.
- You assign weights to the vertices of the Object by painting on it with weight brushes. Starting to paint on a mesh automatically adds weights to the active Vertex Group (a new Vertex Group is created if needed).

Useful Keyboard Shortcuts

The shortcuts can speed up your weight painting:

Weight color picker
: Ctrl LMB change current weight value to the weight value of clicked vertex

Resize the brush
: F then drag to new brush size

Create linear gradient
: Alt LMB then drag

Create radial gradient
: AltCtrl LMB then drag

Draw a *Clipping Border*
: AltB then drag the clipping border to select the part of the 3D window which shall be kept visible. You can then draw only in this part. Press AltB again to remove the *clipping border*.

Weight Paint Mode

Vertex Groups can potentially have a very large number of associated vertices and thus a large number of weights (one weight per assigned vertex). Weight Painting is a method to maintain large amounts of weight information in a very intuitive way. It is primarily used for rigging meshes, where the vertex groups are used to define the relative bone influences on the mesh. But we use it also for controlling particle emission, hair density, many modifiers, shape keys, etc.

The basic principle of the method is: the weight information is literally painted on top of the Mesh body by using a set of Weight brushes. And since painting is always associated with color, we also need to define ...

The weighting Color Code

Weights are visualized by using a cold/hot color system, such that areas of low influence (with weights close to 0.0) are drawn in blue (cold) and areas of high influence (with weights close to 1.0) are drawn in red (hot). And all in-between influences are drawn in rainbow colors, depending on their value (blue, green, yellow, orange, red)

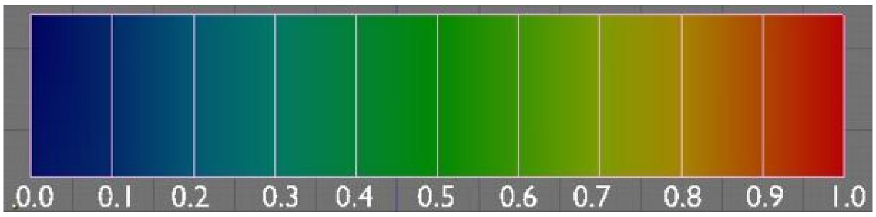
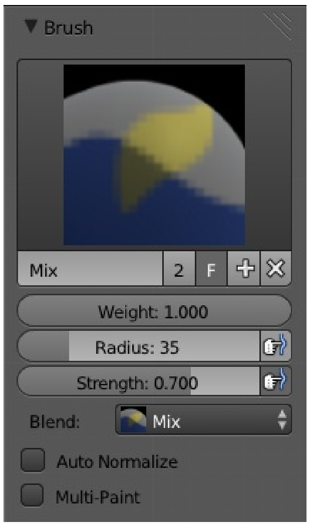


Image 3: The color spectrum and their respective weights.

In addition to the above described color code, Blender has added (as an option) a special visual notation for unreferenced vertices: They are drawn in black. Thus you can see the referenced areas (drawn in cold/hot colors) and the unreferenced areas (in black) at the same time. This is most practical when you look for weighting errors (we will get back to this later).

Brushes



Mix

Weight: 1.000

Radius: 35

Strength: 0.700

Blend: Mix

Auto Normalize

Multi-Paint

The Brush panel in the Tool Shelf

Painting needs paint brushes and Blender provides a Brush Panel within the Tool Shelf when it operates in Weight Paint Mode. You find predefined Brush Presets when you click on the large Brush Icon at the top of the brush Panel. And you can make your own presets as needed. See below for the available brush presets and to create custom presets.

The main brush properties

The most important and frequently modified properties are:

Weight
: The weight (color) to be used by the brush. However, the weight value is applied to the Vertex Group in different ways depending on the selected Brush Blending mode (see below).

Strength
: This is the amount of paint to be applied per brush stroke. What that means exactly also depends on the Brush Blending mode.

Radius
: The radius defines the area of influence of the brush. Note: You can also change the Brush radius with a keyboard shortcut while painting. Just press F at any time, then drag the mouse to increase/reduce the brush radius. Finally click LMB to use the new setting. Or press the Esc key at any time to return to the current settings.

Blend mode
: The brush Blending mode defines in which way the weight value is applied to the Vertex Group while painting. Blender provides 7 different Blending modes:

- **Mix**: In this Blend mode the Weight value defines the target weight that will eventually be reached when you paint long enough on the same location of the mesh. And the strength determines how many strokes you need to arrive at the target weight. Note that for strength = 1.0 the target weight is painted immediately, and for Weight = 0.0 the brush just does nothing.
- **Add**: In this blend mode the specified weight value is added to the vertex weights. The strength determines which fraction of the weight gets added per stroke. However, the brush will not paint weight values above 1.0.
- **Subtract**: In this blend mode the specified weight is subtracted from the vertex weights. The strength determines which fraction of the weight gets removed per stroke. However the brush will not paint weight values below 0.0.
- **Lighten**: In this blend mode the specified weight value is interpreted as the target weight very similar to the Mix Blend mode. But only weights below the target weight are affected. Weights above the target weight remain unchanged.
- **Darken**: This Blend mode is very similar to the Lighten Blend mode. But only weights above the target weight are affected. Weights below the target weight remain unchanged.
- **Multiply**: Multiplies the vertex weights with the specified weight value. This is somewhat like subtract, but the amount of removed weight is now dependent on the Weight value itself.
- **Blur**: tries to smooth out the weighting of adjacent vertices. In this mode the Weight Value is ignored. The strength defines how effectively the blur is applied.

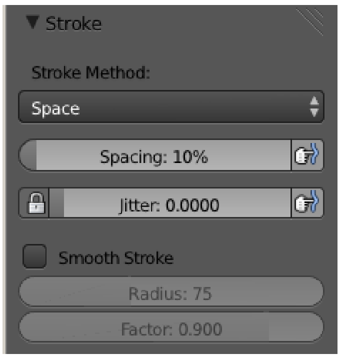
Normalize Options

Blender also provides Options regarding the automatic normalizing of all affected Vertex groups:

Auto Normalize
: Ensures that all deforming vertex groups add up to 1 while painting. When this option is turned off, then all weights of a vertex can have any value between 0.0 and 1.0. However, when Vertex Groups are used as Deform Groups for character animation then Blender always interprets the weight values relative to each other. That is, Blender always does a normalization over all deform bones. Hence in practice it is not necessary to maintain a strict normalization and further normalizing weights should not affect animation at all.

Multi-Paint
: Paint on all selected Vertex Groups simultaneously. This option is only useful in the context of Armatures, where you can select multiple Vertex Groups by selecting multiple Pose bones.

The Brush stroke definition



Stroke Panel

Stroke Method

- **Airbrush**: Keep applying paint effect while holding mouse down (spray)
- **Space**: Limit brush application to the distance specified by spacing (see below)
- **Dots**: Apply paint on each mouse move step

Rate (only for Airbrush)
: Interval between paints for airbrush

Spacing (only for Space)
: Limit brush application to the distance specified by spacing

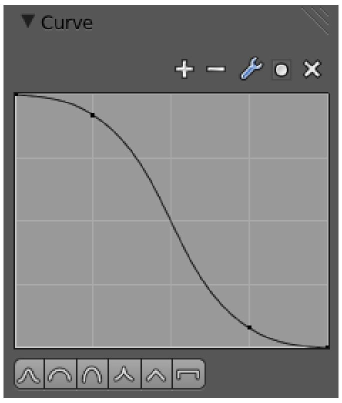
Jitter
: Jitter the position of the brush while painting

Smooth Stroke
: Brush lags behind mouse and follows a smoother path

Radius
: Minimum distance from last point before stroke continues

Factor
: Higher values give a smoother stroke

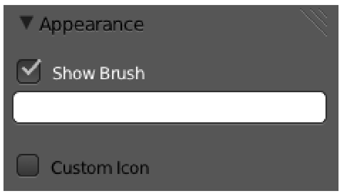
The brush Falloff curve



Curve Panel

The brush falloff editor allows you to speciy the characteristics of your brushes to a large extent. The usage should be obvious and intuitive.

The brush appearance



Brush appearance

Show Brush
: makes the brush visible as a circle (on by default)

Color setter
: To define the color of the brush circle

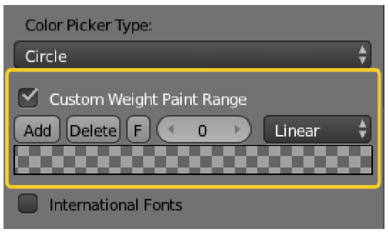
Custom icon
: Allows definition of a custom brush icon

Brush presets

Blender provides several Brush presets:

- **Mix, Draw, Brush**: uses the Mix Blending mode to draw the brush weight with varying strength and brush falloff
- **Add**: uses the Add Blending mode
- **Subtract**: uses the Subtract Blending mode
- **Lighten**: uses the Lighten Blending mode
- **Darken**: uses the Darken Blending mode
- **Multiply**:uses the Multiply Blending mode
- **Blur**: uses the Blur Blending mode

Customizing brush color space

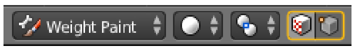


Blender allows customization of the color range used for the Weight Paint colors. You can define the color band as you like; for example, you can make it purely black/white (similar to maya Weight painting), and you can even use Alpha values here.

You find the customizer in the User Properties section, in the System Tab.

Selection Masking

If you have a complex mesh, it is sometimes not easy to paint on all vertices in Weight Paint mode. Suppose you only want to paint on a small area of the Mesh and keep the rest untouched. This is where selection masking comes into play. When this mode is enabled, a brush will only paint on the selected verts or faces. The option is available from the footer menu bar of the 3D viewport (see icons surrounded by the yellow frame):



You can choose between Face Selection masking (left icon) and Vertex selection masking (right icon).

Select mode has some advantages over the default Weight Paint mode:

1. The original mesh edges are drawn, even when modifiers are active.
2. You can select faces to restrict painting to the vertices of the selected faces.
3. Selecting tools include:

Details about selecting

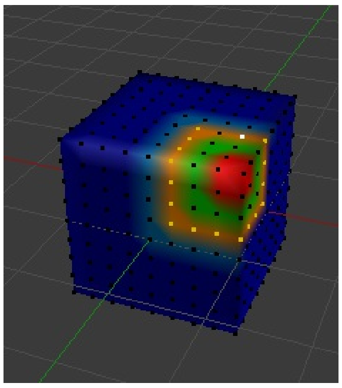
The following standard selection operations are supported:

- RMB – Single faces. Use ⇧ Shift RMB to select multiple.
- A – All faces, also to de-select.
- B – Block/Box selection.
- C – Select with brush.
- L – Pick linked (under the mouse cursor).
- CtrlL – Select linked.
- CtrlI - Invert selection (Inverse).

Selecting Deform Groups

When you are doing weight painting for deform bones (with an Armature), you can select a deform group by selecting the corresponding bone. However, this Vertex Group selection mode is disabled when Selection Masking is active!

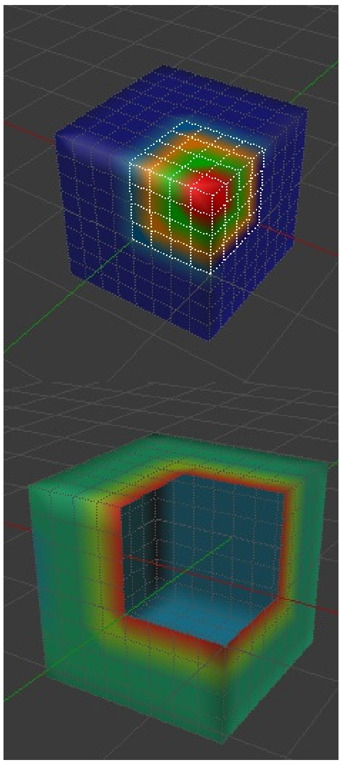
Vertex Selection Masking



In this mode you can select one or more vertices and then paint only on the selection. All unselected vertices are protected from unintentional changes.

Note: This option can also be toggled with the V key:

Face Selection Masking



The Face Selection masking allows you to select faces and limit the weight paint tool to those faces, very similar to Vertex selection masking.

Hide/Unhide Faces

You also can hide selected faces as in Edit Mode with the keyboard Shortcut H, then paint on the remaining visible faces and finally unhide the hidden faces again by using AltH

Hide/Unhide Vertices

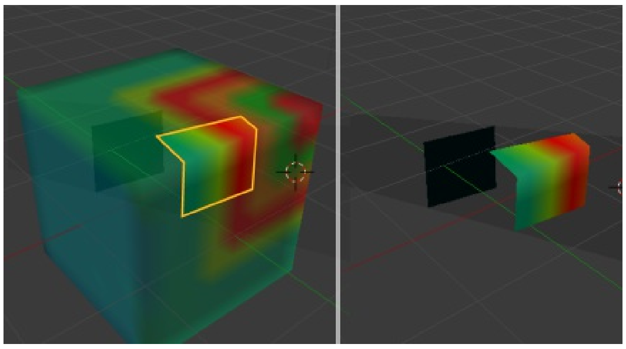
You cannot directly hide selected faces in vertex mask selection mode. However you can use a trick:

1. First go to Face selection mask mode
2. Select the areas you want to hide and then hide the faces (as explained above)
3. Switch back to Vertex Selection mask mode

Now the verts belonging to the hidden Faces will remain hidden.

The Clipping Border

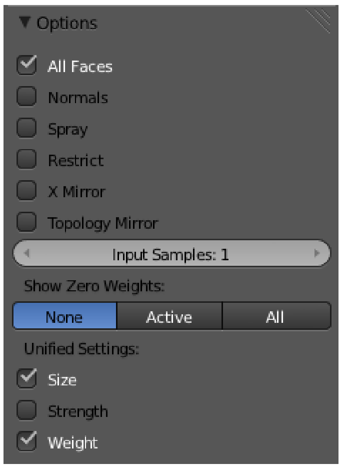
To constrain the paint area further you can use the *Clipping Border*. Press AltB and LMB -drag a rectangular area. The selected area will be "cut out" as the area of interest. The rest of the 3D window gets hidden.



You make the entire mesh visible again by pressing AltB a second time.

All weight paint tools that use the view respect this clipping, including border select, weight gradient and of course brush strokes.

Weight Paint Options



▼ Options

- ✓ All Faces
- Normals
- Spray
- Restrict
- X Mirror
- Topology Mirror

Input Samples: 1

Show Zero Weights: [✓ None] Active All

Unified Settings:

- ✓ Size
- Strength
- ✓ Weight

The Weight Paint Options modify the overall brush behavior:

All faces
: If this is turned off, you will only paint vertices which belong to a face on which the cursor is located. This is useful if you have a complicated mesh and you might paint on visually near faces that are actually quite distant in the mesh.

Normals
: The vertex normal (helps) determine the extent of painting. This causes an effect as if painting with light.

Spray
: The Spray option accumulates weights on every mouse move.

Restrict
: The Restrict option limits the influence of painting to vertices belonging (even with weight 0) to the selected vertex group.

X-mirror
: Use the X-mirror option for mirrored painting on groups that have symmetrical names, like with extension `.R`/`.L`, or `_R`/`_L`. If a group has no mirrored counterpart, it will paint symmetrically on the active group itself. You can read more about the naming convention in Editing Armatures: Naming conventions. The convention for armatures/bones apply here as well.

Topology Mirror
: Use topology-based mirroring, for when both side of a mesh have matching mirrored topology.

Input Samples
: ...

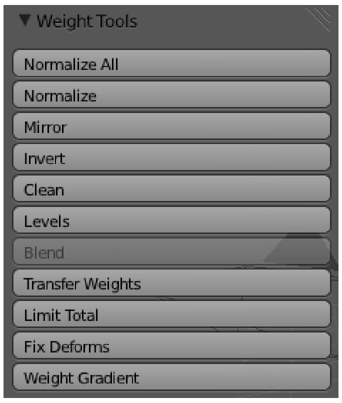
Show Zero Weights

- None
- Active
- All

Unified Settings: The Size, Strength and Weight of the brush can be set to be shared across different brushes, as opposed to per-brush.

- Spray: to constantly draw (opposed to drawing one stroke per mouse click).
- Restrict: to only paint on vertices which already are weighted in the active weight group. (No new weights are created; only existing weights are modified.)
- x-mirror: to draw symmetrically. Note the this only works when the character symmetry plane is z-y (character looks into y direction).
- Show Zero weights: To display unreferenced and zero weighted areas in black (by default).

Weight Paint Tools



Blender provides a set of helper tools for Weight Painting. The tools are located in the weight tools panel.

The weight paint tools are full described in the Weight Paint Tools page

Weight Painting for Bones

This is probably the most often used application of weight painting. When a bone moves, vertices around the joint should move as well, but just a little, to mimic the stretching of the skin around the joint. Use a “light” weight (10-40%) paint on the vertices around the joint so that they move a little when the bone rotates. While there are ways to automatically assign weights to an armature (see the Armature section), you can do this manually. To do this from scratch, refer to the process below. To modify automatically assigned weights, jump into the middle of the process where noted:

- Create an armature.
- Create a mesh that will be deformed when the armature’s bone(s) move.
- With the mesh selected, create an Armature modifier for your mesh (located in the Editing context, Modifiers panel). Enter the name of the armature.

Pick up here for modifying automatically assigned weights.

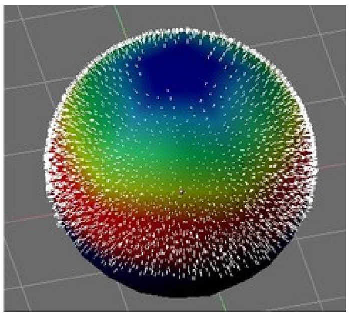
- Select the armature in 3D View, and bring the armature to **Pose mode** (Ctrl⇆ Tab, or the 3D View window header mode selector).
- Select a desired bone in the armature.
- Select your mesh (using RMB) and change immediately to Weight Paint mode. The mesh will be colored according to the weight (degree) that the selected bone movement affects the mesh. Initially, it will be all blue (no effect).
- Weight paint to your heart’s content. The mesh around the bone itself should be red (generally) and fade out through the rainbow to blue for vertices farther away from the bone.

You may select a different bone with RMB while weight painting, provided the armature was left in Pose mode as described above. This will activate the vertex group sharing the name with the selected bone, and display related weights. If the mesh skins the bones, you will not be able to see the bones because the mesh is painted. If so, turn on X-Ray view (Buttons window, Editing context, Armature panel). While there on that panel, you can also change how the bones are displayed (Octahedron, Stick, B-Bone, or Envelope) and enable Draw Names to ensure the name of the selected bone matches up to the vertex group.

If you paint on the mesh, a vertex group is created for the bone. If you paint on vertices outside the group, the painted vertices are automatically added to the vertex group.

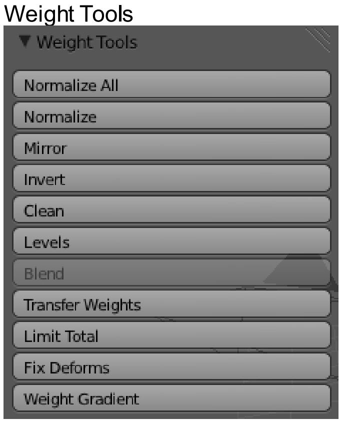
If you have a symmetrical mesh and a symmetrical armature you can use the option X-Mirror. Then the mirrored groups with the mirrored weights are automatically created.

Weight Painting for Particles


Weight painted particle emission.

Faces or vertices with zero weight generate no particles. A weight of 0.1 will result in 10% of the amounts of particles. This option “conserves” the total indicated number of particles, adjusting the distributions so that the proper weights are achieved while using the actual number of particles called for. Use this to make portions of your mesh hairier than others by weight painting a vertex group, and then calling out the name of the vertex group in the VGroup: field (Particles panel, Object context).

Weight Tools



Blender provides a set of helper tools for Weight Painting. The tools are accessible from the Tool Shelf in Weight Paint mode. And they are located in the weight tools panel.

The Subset Option

Some of the tools also provide a Subset parameter (in the Operator panel, displayed after the tool is called) with following options:

- Active Group
- Selected Pose Bones
- Deform pose Bones
- All Groups

All tools also work with Vertex Selection Masking and Face Selection masking. In these modes the tools operate only on selected verts or faces.

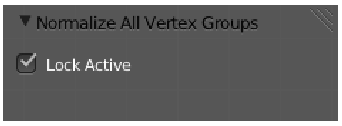
About the Blend tool

The Blend tool only works when “Vertex selection masking for painting” is enabled. Otherwise the tool button is grayed out.

Normalize All

For each vertex, this tool makes sure that the sum of the weights across all Vertex Groups is equal to 1. This tool normalizes all of the vertex groups, except for locked groups, which keep their weight values untouched.

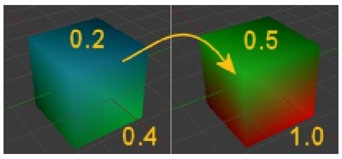
Operator parameters



Lock Active
: Keep the values of the active group while normalizing all the others.

Please note: Currently this tool normalizes ALL vertex groups except the locked vertex groups.

Normalize

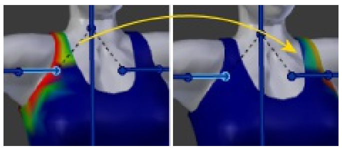


This tool only works on the active Vertex Group. All vertices keep their relative weights, but the entire set of weights is scaled up such that the highest weight value is 1.0

Operator parameters

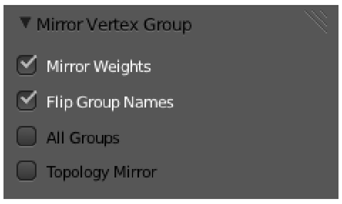
None

Mirror



This tool mirrors the weights from one side of the mesh to the opposite side (only mirroring along x-axis is supported). But note, the weights are not transferred to the corresponding opposite bone weight group. The mirror only takes place within the selected Vertex Group.

Operator parameters



Mirror Weights
: Mirrors the weights of the active group to the other side. Note, this only affects the active weight group.

Flip Group Names
: Exchange the names of left and right side. This option only renames the groups.

All Groups
: Operate on all selected bones.

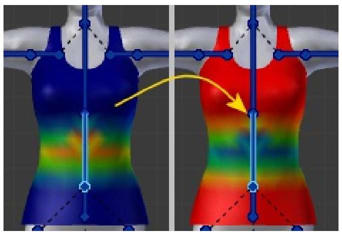
Topology Mirror
: Mirror for meshes which are not 100% symmetric (approximate mirror).

Mirror to opposite bone

If you want to create a mirrored weight group for the opposite bone (of a symmetric character), then you can do this:

- Delete the target Vertex Group (where the mirrored weights will be placed)
- Create a copy of the source bone Vertex Group (the group containing the weights which you want to copy)
- Rename the new Vertex Group to the name of the target Vertex Group (the group you deleted above)
- Select the Target Vertex Group and call the Mirror tool (use only the Mirror weights option and optionally Topology Mirror if your mesh is not symmetric)

Invert



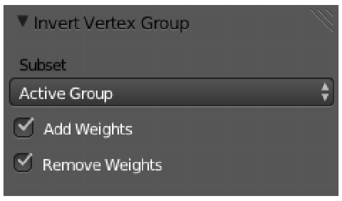
Replaces each Weight of the selected weight group by 1.0 – weight.

Examples:

- original 1.0 converts to 0.0
- original 0.5 remains 0.5
- original 0.0 converts to 1.0

Note: Please see how the parameter settings change the behavior.

Operator parameters



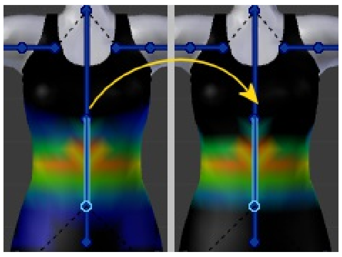
Subset
: Restrict the tool to a subset. See above (The Subset Option) about how subsets are defined.

Add Weights
: Add verts that have no weight before inverting (these weights will all be set to 1.0)

Remove Weights
: Remove verts from the Vertex Group if they are 0.0 after inverting.

Note: Locked vertex Groups are not affected.

Clean

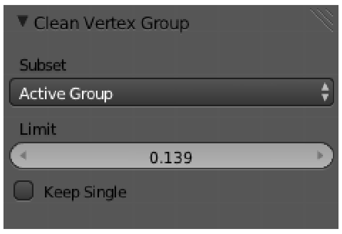


Removes weights below a given threshold. This tool is useful for clearing your weight groups of very low (or zero-) weights.

In the example shown, I used a cutoff value of 0.139 (see operator options below) so all blue parts (left side) are cleaned out (right side).

Note, the images use the Show Zero weights=Active option so that unreferenced Weights are shown in Black.

Operator parameters



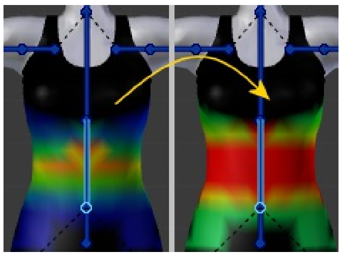
▼ Clean Vertex Group
Subset: Active Group
Limit: 0.139
☐ Keep Single

Subset
: Restrict the tool to a subset. See above (The Subset Option) for how subsets are defined.

Limit
: This is the minimum weight value that will be kept in the Group. Weights below this value will be removed from the group.

Keep Single
: Ensure that the Clean tool will not create completely unreferenced verts (verts which are not assigned to any Vertex Group), so each vertex will keep at least one weight, even if it is below the limit value!

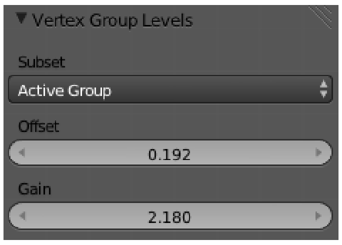
Levels



Adds an offset and a scale to all weights of the selected Weight Groups. with this tool you can raise or lower the overall “heat” of the weight group.

Note: No weight will ever be set to values above 1.0 or below 0.0 regardless of the settings.

Operator parameters



▼ Vertex Group Levels
Subset: Active Group
Offset: 0.192
Gain: 2.180

Subset
: Restrict the tool to a subset. See above (The Subset Option) for how subsets are defined.

Offset

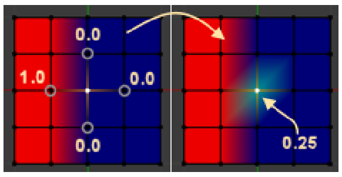
A value from the range [-1.0,1.0]) to be added to all weights in the Vertex Group.

Gain
: All weights in the Subset are multiplied with the gain. The drag sliders of this value allow only a range of [-10.0, 10.0]. However, you can enter any factor you like here by typing from the keyboard.

Note: Whichever Gain and Offset you choose, in all cases the final value of each weight will be clamped to the range [0.0, 1.0]. So you will never get negative weights or overheated areas (weight > 1.0) with this tool.

Blend

Blends the weights of selected vertices with adjacent unselected vertices. This tool only works in vertex select mode.

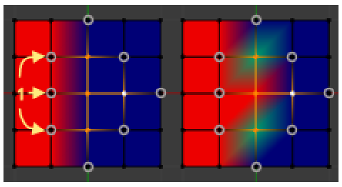


To understand what the tool really does, let's take a look at a simple example. The selected vertex is connected to 4 adjacent vertices (marked with a gray circle in the image). All adjacent vertices are unselected. Now the tool calculates the average weight of all connected **and** unselected verts. In the example this is

(1 + 0 + 0 + 0) / 4 = 0.25

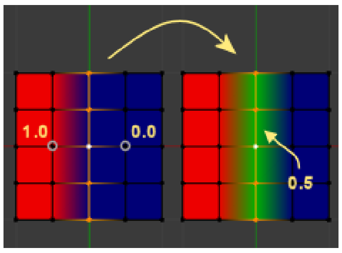
This value is multiplied by the factor given in the Operator parameters (see below).

- If the factor is 0.0 then actually nothing happens at all and the vertex just keeps its value.
- If the factor is 1.0 then the calculated average weight is taken (0.25 here).
- Dragging the factor from 0 to 1 gradually changes from the old value to the calculated average.



Now let's see what happens when we select all but one of the neighbors of the selected vert as well. Again all connected and unselected verts are marked with a gray circle. When we call the Blend tool now and set the Factor to 1.0, then we see different results for each of the selected verts:

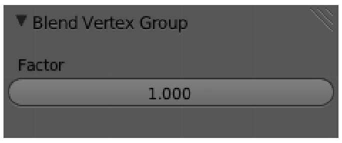
- The topmost and bottommost selected verts:
  are surrounded by 3 unselected verts, with an average weight of (1 + 0 + 0) / 3 = 0.333 So their color has changed to light green.
- The middle vertex:
  is connected to one unselected vert with weight = 1. So the average weight is 1.0 in this case, thus the selected vert color has changed to red.
- The right vert:
  is surrounded by 3 unselected verts with average weight = (0+0+0) / 3 = 0 So the average weight is 0, thus the selected vert color has not changed at all (it was already blue before blend was applied).



Finally let's look at a practical example (and explain why this tool is named Blend). In this example I have selected the middle edge loop. And I want to use this edge loop for blending the left side to the right side of the area.

- All selected vertices have 2 unselected adjacent verts.
- The average weight of the unselected verts is (1 + 0) / 2 = 0.5
- Thus when the Blend Factor is set to 1.0 then the edge loop turns to green and finally does blend the cold side (right) to the hot side (left).

Operator parameters

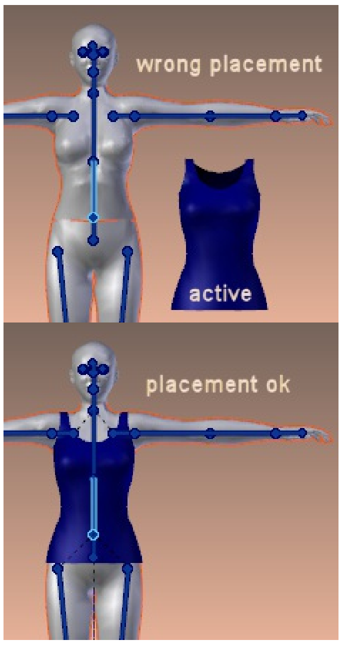


Factor
: The effective amount of blending (range [0.0, 1.0]). When Factor is set to 0.0 then the Blend tool does not do anything. For Factor > 0 the weights of the affected vertices gradually shift from their original value towards the average weight of all connected **and** unselected verts (see examples above).

Transfer Weights

Copy weights from other objects to the vertex groups of the active Object. By default this tool copies all vertex groups contained in the selected objects to the target object. However you can change the tool's behavior in the operator redo panel (see below).

Prepare the copy



You first select all source objects, and finally the target object (the target object must be the active object).

It is important that the source objects and the target object are at the same location. If they are placed side by side, then the weight transfer won't work. You can place the objects on different layers, but you have to ensure that all objects are visible when you call the tool.

Now ensure that the Target Object is in Weight Paint mode.

Call the tool

Open the Tool Shelf and locate the Weight Tools panel. From there call the "Transfer weights" tool. The tool will initially copy all vertex groups from the source objects. However the tool also has an operator redo panel (which appears at the bottom of the tool shelf). From the redo panel you can change the parameters to meet your needs. (The available Operator parameters are documented below.)

Redo Panel Confusion

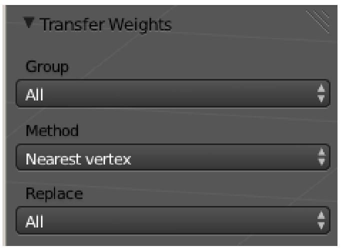
You may notice that the Operator Redo Panel (see below) stays available after the weight transfer is done. The panel only disappears when you call another Operator that has its own redo Panel. This can lead to confusion when you use Transfer weights repeatedly after you changed your vertex groups. If you then use the still-visible redo panel, then Blender will reset your work to its state right before you initially called the Transfer Weights tool.

Workaround

When you want to call the Transfer Weights tool again after you made some changes to your vertex groups, then always use the "Transfer Weights" Button, even if the operator panel is still available. Unless you really want to reset your changes to the initial call of the tool.

Operator parameters

Defaults are marked in boldface:



Group

- Active: Only copy to the Active Group in the active Object. This option only works when the active Object has an active Vertex Group set. Otherwise the Weight transfer will not do anything.
- **All**: Copy all Vertex groups from the selected objects to the Active Object.

Method

- **Nearest vertex In face**: to be documented
- Nearest Face: to be documented
- Nearest vertex: to be documented
- Vertex Index (verbatim copy, works only for meshes with identical index count)

Replace

- Empty: Only copy a weight to the active object if the vertex has not yet had a weight set in the group.
- **All**: delete all previous content of the target vertex group before copying the group from the source object.

Caveat!

If a vertex group is contained in 2 or more of the selected objects, then the result depends on the order in which the selected objects are processed. However, the order of processing cannot be influenced.

Limit total

Reduce the number of weight groups per vertex to the specified Limit. The tool removes lowest weights first until the limit is reached.

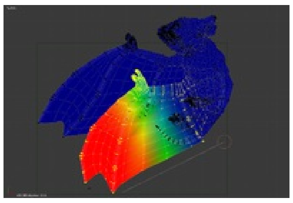
Hint: The tool can only work reasonably when more than one weight group is selected.

Operator parameters

Subset
: Restrict the tool to a subset. See above (The Subset Option) for how subsets are defined.

Limit
: Maximum number of weights allowed on each vertex (default:4)

Weight Gradient (wip)


example of the gradient tool being used with selected vertices.

This is an interactive tool for applying a linear/radial weight gradient; this is useful at times when painting gradual changes in weight becomes difficult.

The gradient tool can be accessed from the Toolbar as a key shortcut:

- Linear: Alt LMB and drag
- Radial: AltCtrl LMB and drag

The following weight paint options are used to control the gradient:

- Weight - The gradient starts at the current selected weight value, blending out to nothing.
- Strength - Lower values can be used so the gradient mixes in with the existing weights (just like with the brush).
- Curve - The brush falloff curve applies to the gradient too, so you can use this to adjust the blending.

Blends the weights of selected vertices with unselected vertices. Hint: this tool only works in vertex select mode.

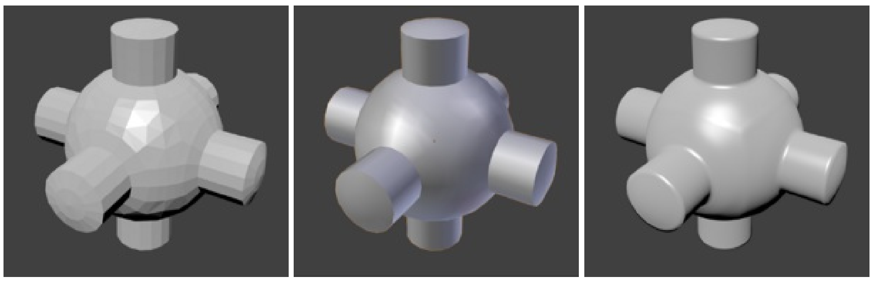
Operator parameters

Type

- Linear
- Radial

X Start
X End
Y Start
Y End

Mesh Shading


Example mesh rendered flat, smoothed using edge split, and using Subdivision Surface. Note how edges are rendered differently. Sample .blend

As seen in the previous sections, polygons are central to Blender. Most objects are represented by polygons and truly curved objects are often approximated by polygon meshes. When rendering images, you may notice that these polygons appear as a series of small, flat faces.

Sometimes this is a desirable effect, but usually we want our objects to look nice and smooth. This section shows you how to visually smooth an object, and how to apply the Auto Smooth filter to quickly and easily combine smooth and faceted polygons in the same object.

The last section on this page shows possibilities for smoothing a mesh’s geometry, not only its appearance.

Smooth shading

Mode: Edit and Object mode

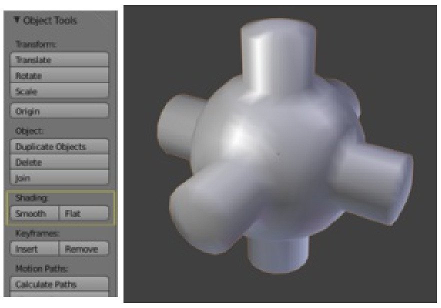
Panel: Mesh Tools (Editing context)

Hotkey: CtrlF » Shade Smooth / Shade Flat

Menu: Mesh » Faces » Shade Smooth / Shade Flat

The easiest way is to set an entire object as smooth or faceted by selecting a mesh object, and in Object mode, click Smooth in the Tool Shelf. This button does not stay pressed; it forces the assignment of the “smoothing” attribute to each face in the mesh, including when you add or delete geometry.

Notice that the outline of the object is still strongly faceted. Activating the smoothing features doesn’t actually modify the object’s geometry; it changes the way the shading is calculated across the surfaces, giving the illusion of a smooth surface. Click the Flat button in the Tool Shelf's Shading panel to revert the shading back to that shown in the first image above.



Same mesh smooth shaded

Smoothing parts of a mesh

Alternatively, you can choose which edges to smooth by entering Edit mode, then selecting some faces and clicking the Smooth button. The selected edges are marked in yellow.

When the mesh is in Edit mode, only the selected edges will receive the “smoothing” attribute. You can set edges as flat (removing the “smoothing” attribute) in the same way by selecting edges and clicking the Flat button.

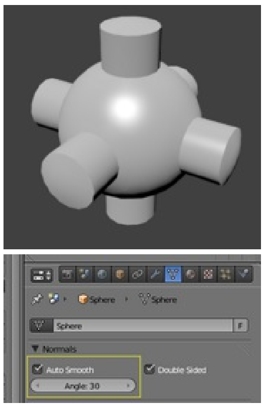
Auto Smooth

Panel: Properties (Object Data context)

It can be difficult to create certain combinations of smooth and solid faces using the above techniques alone. Though there are workarounds (such as splitting off sets of faces by selecting them and pressing Y), there is an easier way to combine smooth and solid faces, by using Auto Smooth.

Auto smoothing can be enabled in the mesh's panel in the Properties window. Angles on the model that are smaller than the angle specified in the Angle button will be smoothed when that part of the mesh is set to smooth. Higher values will produce smoother faces, while the lowest setting will look identical to a mesh that has been set completely solid.

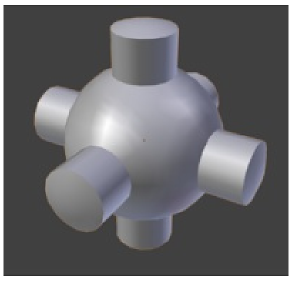
Note that a mesh, or any faces that have been set as Flat, will not change their shading when Auto Smooth is activated: this allows you extra control over which faces will be smoothed and which ones won’t by overriding the decisions made by the Auto Smooth algorithm.

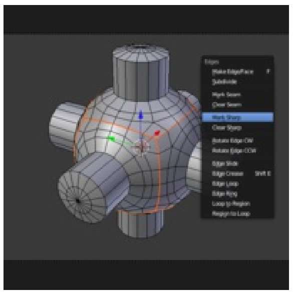


Example mesh with Auto Smooth enabled

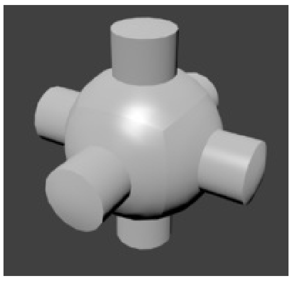
Edge Split Modifier

With the Edge Split Modifier we get a result similar to Auto Smooth with the ability to choose which edges should be split, based on angle—those marked as sharp.


Edge Split modifier enabled, based on angle



Edges marked as sharp


Resulting render with sharp edge weighting

Smoothing the mesh geometry

The above techniques do not alter the mesh itself, only the way it is displayed and rendered. Instead of just making the mesh look like a smooth surface, you can also physically smooth the geometry of the mesh with these tools:

Mesh editing tools

You can apply one of the following in Edit mode:

Smooth
: This relaxes selected components, resulting in a smoother mesh.

Laplacian Smooth
: Smooths geometry by offers controls for better preserving larger details.

Subdivide Smooth
: Adjusting the smooth parameter after using the subdivide tool results in a more organic shape. This is similar to using the subdivide modifier.

Bevel
: This Bevels selected edged, causing sharp edges to be flattened.

Modifiers

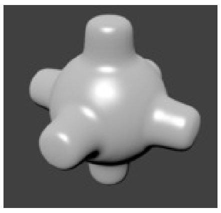
Alternatively, you can smooth the mesh non-destructively with one or several of the following modifiers:

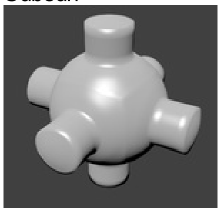
Smooth Modifier
: Works like the Smooth tool in Edit mode; can be applied to specific parts of the mesh using vertex groups.

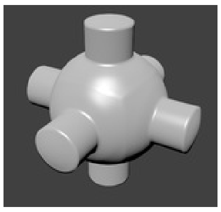
Laplactian Smooth Modifier
: Works like the Laplacian Smooth tool in Edit mode; can be applied to specific parts of the mesh using vertex groups.

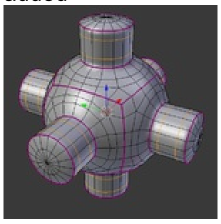
Bevel Modifier
: Works like the Bevel tool in Edit mode; Bevel can be set to work on an angle threshold, or on edge weight values.

Subdivision Surface Modifier
: Catmull-Clark subdivision produces smooth results. Sharp edges can be defined with subdivision creases or by setting certain edges to “sharp” and adding an EdgeSplit modifier (set to From Marked As Sharp) before the Subsurf modifier.


Subsurf


Using creased edges, and resulting subsurf artifacts


Extra edge loops added


3D view showing creased edges (pink) and added edges loops (yellow)

Mesh Clean-up

These tools are to help cleanup degenerate geometry and fill in missing areas of a mesh.

Fill Holes

Mode: Edit mode

Menu: Mesh » Clean up » Fill Holes

This tool is can take a large selection and detect the holes in the mesh, filling them in.

This is different from the face creation operator in three important respects.

- holes are detected, so there is no need to manually find and select the edges around the holes.
- holes can have a limit for the number of sides (so only quads or tris are filled in for example).
- mesh data is copied from surrounding geometry (UV's, vertex-colors, multi-res, all layers), since manually creating this data is very time consuming.

Split Non-Planar Faces

Mode: Edit mode

Menu: Mesh » Clean up » Split Non-Planar Faces

This tool avoids ambiguous areas of geometry by splitting non-flat faces when they are bent beyond a given limit.

Delete Loose Geometry

Mode: Edit mode

Menu: Mesh » Clean up » Delete Loose

This tool removes disconnected vertices and edges (optionally faces - off by default).

Degenerate Dissolve

Mode: Edit mode

Menu: Mesh » Clean up » Degenerate Dissolve

This tool collapses / removes geometry which you typically wont want.

- Edges with no length.
- Faces with no areas (faces on a point or thin faces).
- Face corners with no area.

Curves

Bird logo made from
Bezier curves.

Curves and Surfaces are particular types of Blender Objects. They are expressed by mathematical functions rather than a series of points. Blender offers both Bezier curves and NURBS curves and surfaces. NURBS stands for "Non-Uniform Rational B-Splines". Both Bezier curves and NURBS curves and surfaces are defined in terms of a set of “control points” (or “control vertices”) which define a “control polygon”.

The main advantage to using curves instead of polygonal meshes is that curves are defined by less data and so can produce excellent results using less memory and storage space at modeling time. However, this procedural approach to surfaces can increase demands at render time.

Certain modeling techniques, such as extruding a profile along a path, are possible only using curves. On the other hand, when using curves, vertex-level control is more difficult and if fine control is necessary, mesh editing may be a better modeling option.

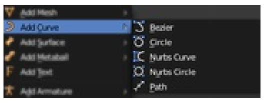
Bezier curves are the most commonly used curves for designing letters or logos. They are also widely used in animation, both as paths for objects to move along and as F-curves to change the properties of objects as a function of time.

Tutorials

Create the bird logo with Bezier Curves »

Skinning: Making a surface with two or more curves »

Curve Primitives



Add Curve menu.

In Object mode, the Add Curve menu, Blender provides five different curve primitives:

Bezier Curve
: Adds an open 2D Bezier curve with two control points.

Bezier Circle
: Adds a closed, circle-shaped 2D Bezier curve (made of four control points).

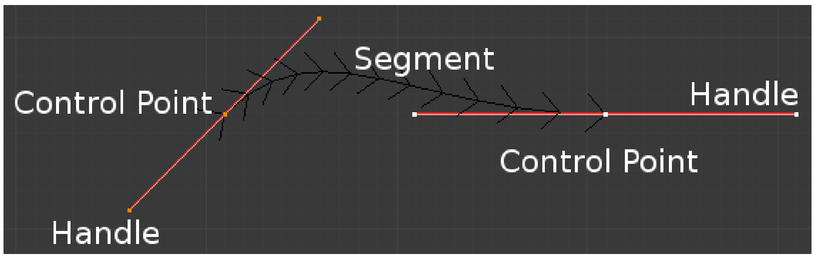
NURBS Curve
: Adds an open 2D NURBS curve, with four control points, with Uniform knots.

NURBS Circle
: Adds a closed, circle-shaped 2D NURBS curve (made of eight control points).

Path
: Adds a NURBS open 3D curve made of five aligned control points, with Endpoint knots and the CurvePath setting enabled.

Bezier Curves

The main elements used in editing Bezier Curves are the Control Points and Handles. A Segment (the actual Curve) is found between two Control Points. In the image below, the Control Points can be found in the middle of the pink line while the Handles comprise the extensions from the Control Point. By default the arrows on the Segment represents the direction and **relative** speed and direction of movement Objects will have when moving along the curve. This can be altered by defining a custom Speed Ipo.



Bezier Curve in Edit mode.

Editing Bezier Curves

A Bezier curve can be edited by moving the locations of the Control Points and Handles.

1. Add a Curve by ⇧ ShiftA to bring up the Add menu, followed by Curve » Bezier.
2. Press ⇆ Tab to enter Edit mode.
3. Select one of the Control Points and move it around. Use LMB to confirm the new location of the Control Point, or use RMB

to cancel.
4. Now select one of the Handles and move it around. Notice how this changes the curvature of the curve.

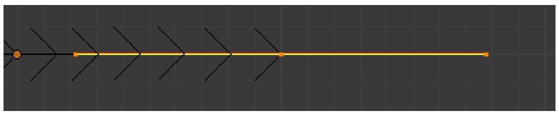
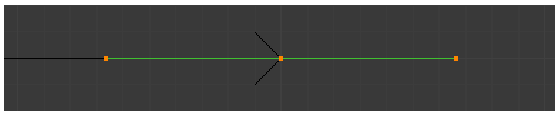
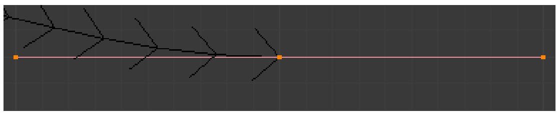
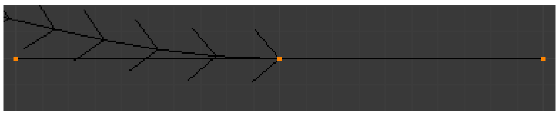
To add more Control Points

1. Select at least two adjacent Control Points.
2. Press W and select Subdivide.
3. Optionally, you can press F6 immediately after the subdivision to modify the number of subdivisions.

Note that while in Edit mode you cannot directly select a Segment. To do so, select all of the Control Points that make up the Segment you want to move.

There are four Bezier curve handle types. They can be accessed by pressing V and selecting from the list that appears, or by pressing the appropriate hotkey combination. Handles can be rotated, moved, scaled and shrunk/fattened like any vertex in a mesh.

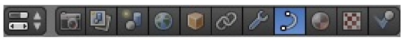
Bezier Curve Handle Types

Type	Shortcut	Usage	Appearance
Automatic	VA	This handle has a completely automatic length and direction which is set by Blender to ensure the smoothest result. These handles convert to Aligned handles when moved.	
Vector	VV	Both parts of a handle always point to the previous handle or the next handle which allows you to create curves or sections thereof made of straight lines or with sharp corners. Vector handles convert to Free handles when moved.	
Aligned	VL	These handles always lie in a straight line, and give a continuous curve without sharp angles.	
Free	VF	The handles are independent of each other.	

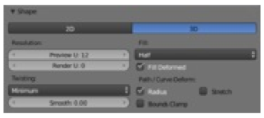
Additionally, the VT shortcut can be used to toggle between Free and Aligned handle types.

Curve Properties

Curve Properties can be set from the Object Data option in the Properties Header (shown below in blue).



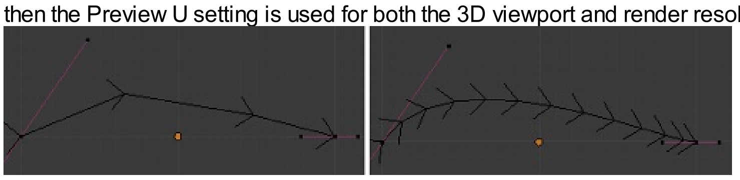
Shape



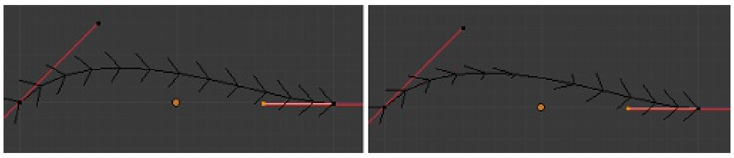
Curves Shape panel.

2D and 3D Curves
: By default, new curves are set to be 3D, which means that Control Points can be placed anywhere in 3D space. Curves can also be set to 2D which constrain the Control Points to the Curve's local XY axis.

Resolution
: The *resolution* property defines the number of points that are computed between every pair of Control Points. Curves can be made more or less smooth by increasing and decreasing the resolution respectively. The Preview U setting determines the resolution in the 3D viewport while the Render U setting determines the Curve's render resolution. If Render U is set to zero (0), then the Preview U setting is used for both the 3D viewport and render resolution.

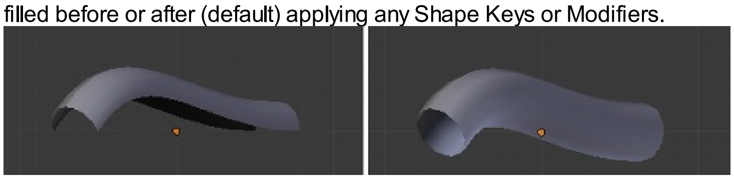

Curves with a resolution of 3 (left) and 12 (right).

Twisting
: A 3D Curve has Control Points that are not located on the Curve's local XY plane. This gives the Curve a twist which can affect the Curve normals. You can alter how the twist of the Curve is calculated by choosing from Minimum, Tangent and Z-Up options from the drop-down menu.


Curves with a twist of *minimum* (left) and *tangent* (right).

Fill

Fill determines the way a Curve is displayed when it is Beveled (see below for details on Beveling). When set to Half (the default) the Curve is displayed as half a cylinder. The Fill Deformed option allows you to indicate whether the Curve should be filled before or after (default) applying any Shape Keys or Modifiers.

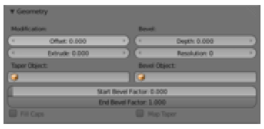

Curves with a fill of *half* (left) and *full* (right).

Path/Curve-Deform

These options are primarily utilized when using a Curve as a Path or when using the Curve Deform property. The Radius, Stretch and Bounds Clamp options control how Objects use the Curve and are dealt with in more detail in the appropriate links below.

Read more about Basic Curve Editing »
Read more about Paths »
Read more about Curve Deform »

Geometry



Curves Geometry panel.

Modification

Offset

By default, text Objects are treated as curves. The Offset option will alter the space between letters.

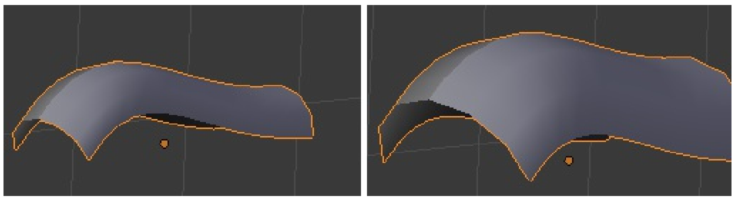
Extrude

Will extrude the curve along both the positive and negative local Z axes.

Bevel

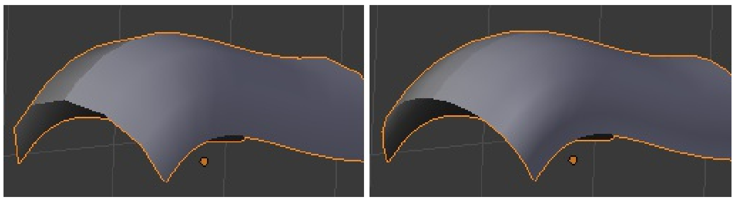
Depth

Changes the size of the bevel


A Curve with different Bevel depths applied.

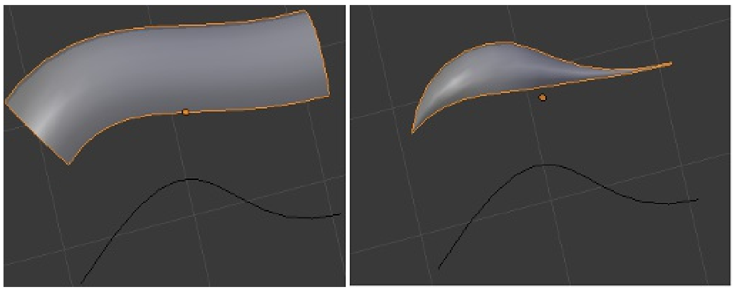
Resolution

Alters the smoothness of the bevel

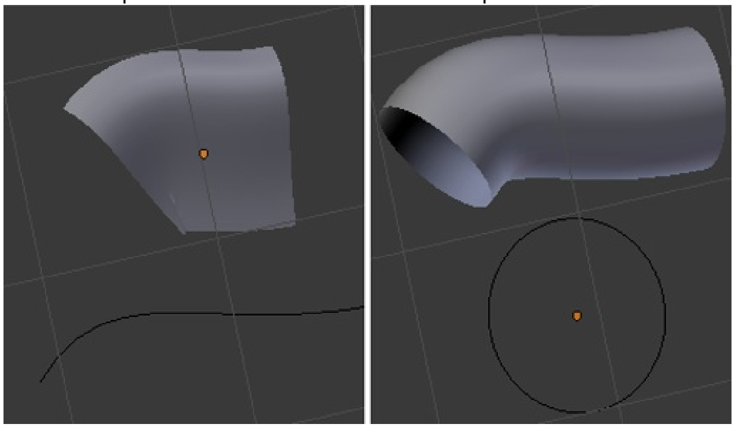

A Curve with different resolutions applied.

Taper Object

Tapering a Curve causes it to get thinner towards one end. You can also alter the proportions of the Taper throughout the tapered object by moving/scaling/rotating the Control Points of the Taper Object. The Taper Object can only be another Curve. Editing the Handles and Control Points of the Taper Object will cause the original Object to change shape.

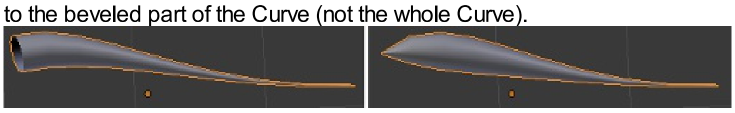

A Curve before (left) and after (right) a Bezier Curve Taper Object was applied.

Bevel Object
: Beveling a Bezier Curve with a Bezier Curve as the Bevel Object generally gives it the appearance of a plane, while using a Bezier Circle as the Bevel Object will give it the appearance of a cylinder. The Bevel Object can only be another Curve. Editing the Handles and Control Points of the Bevel Object will cause the original Object to change shape. Given the options available, it is best to experiment and see the results of this operation.

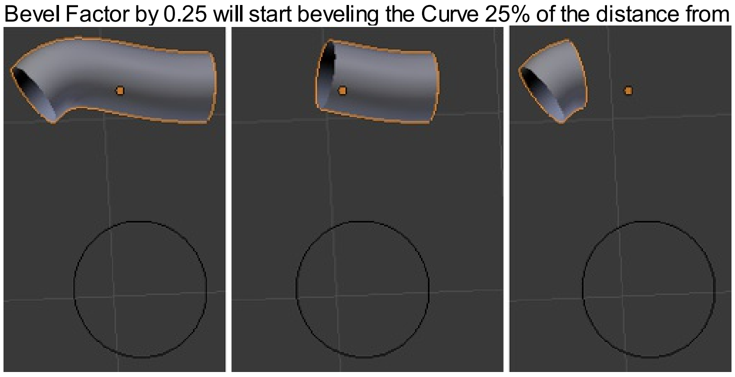

A Curve with the Bevel Object as a Bezier Curve (left) and as a Bezier Circle (right).

Fill Caps
: Seals the ends of a beveled Curve.

Map Taper
: For Curves using a Taper Object and with modifications to the Start/End Bevel Factor the Map Taper option will apply the taper to the beveled part of the Curve (not the whole Curve).


A Curve without (left) and with (right) Map Taper applied.

Start Bevel Factor and End Bevel Factor
: These options determine where to start the Bevel operation on the Curve being beveled. Increasing the Start Bevel Factor to 0.5 will start beveling the Curve 50% of the distance from the start of the Curve (in effect shortening the Curve). Decreasing the End Bevel Factor by 0.25 will start beveling the Curve 25% of the distance from the end of the Curve (again, shortening the Curve).


A Curve with no Bevel factor applied (left), with a 50% Start Bevel Factor (middle) and with a 25% End Bevel Factor (right).

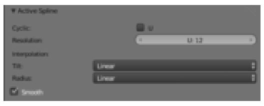
Read more about Advanced Curve Editing »

Path Animation

The Path Animation settings can be used to determine how Objects move along a certain path. See the link below for further information.

Read more about utilizing Curves for paths during animation »

Active Spline



Curves Active Spline panel.

The Active Spline panel becomes available during Edit mode.

Cyclic
: Closes the Curve.

Resolution
: Alters the smoothness of of each segment by changing the number of subdivisions.

Interpolation
: Tilt

    Alters how the tilt of a segment is calculated.

  Radius

    Alters how the radius of a Beveled Curve is calculated. The effects are easier to see after Shrinking/Fattening a control point AltS.

  Smooth

    Smooths the normals of the Curve

Non-Uniform Rational B-Splines (NURBS)

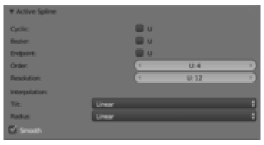
One of the major differences between Bezier Objects and NURBS Objects is that Bezier Curves are approximations. For example, a Bezier circle is an *approximation* of a circle, whereas a NURBS circle is an *exact* circle. NURBS theory can be a *very* complicated topic. For an introduction, please consult the Wikipedia page. In practice, many of the Bezier curve operations discussed above apply to NURBS curves in the same manner. The following text will concentrate only on those aspects that are unique to NURBS curves.

Editing NURBS Curves

A NURBS Curve is edited by moving the location of the Control Points.

1. Place a Curve by ⇧ ShiftA to bring up the Add menu, followed by Curve » NURBS curve.
2. Press ⇆ Tab to enter Edit mode.
3. Select one of the Control Points and move it around. Use LMB to confirm the new location of the Control Point, or use RMB to cancel.
4. If you want to add additional Control Points, select both of them, press W and select Subdivide. Press F6 immediately after to determine how many subdivisions to make.

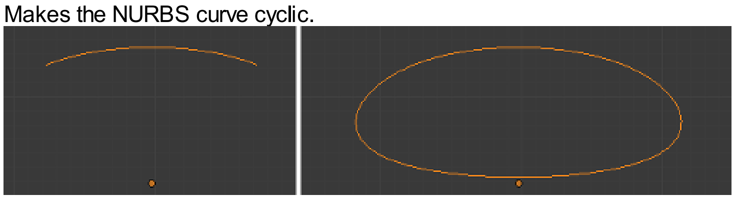
Active Spline



NURBS Active Spline panel.

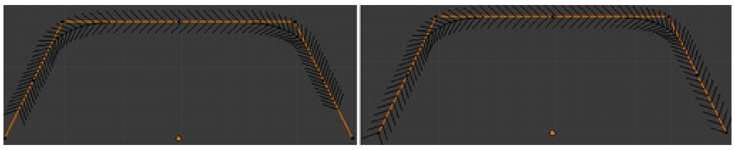
One of the characteristics of a NURBS object is the *knot vector*. This is a sequence of numbers used to determine the influence of the control points on the curve. While you cannot edit the knot vectors directly, you can influence them through the Endpoint and Bezier options in the Active Spline panel. Note that the Endpoint and Bezier settings only apply to open NURBS curves.

Cyclic
: Makes the NURBS curve cyclic.

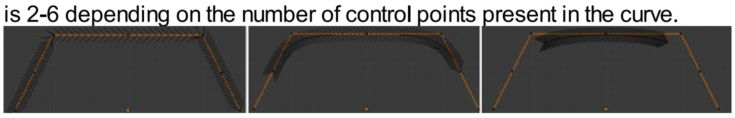

A NURBS curve with Cyclic applied.

Bezier
: Makes the NURBS curve act like a Bezier curve.

Endpoint
: Makes the curve contact the end control points. Cyclic must be disabled for this option to work.


A NURBS curve with Endpoint enabled.

Order
: The order of the NURBS curve determines the area of influence of the control points over the curve. Higher order values means that a single control point has a greater influence over a greater relative proportion of the curve. The valid range of Order values is 2-6 depending on the number of control points present in the curve.


NURBS curves with orders of 2 (left), 4 (middle) and 6 (right).

Path

As mentioned above, Curves are often used as paths. Any curve can be used as a Path if the Path Animation option is selected.

The Path option available from the Add Curve menu is identical to a 3D NURBS curve, except that you do not have access to the Active Spline panel.

Curve Selection

Curve selection in Edit mode is much less complex than with meshes! Mainly this is because you have only one selectable element type, the control points (no select mode needed here…). These points are a bit more complex than simple vertices, however, especially for Béziers, as there is the central vertex, and its two handles…

The basic tools are the same as with meshes, so you can select a simple control point with a LMB -click, add to current selection with ⇧ Shift LMB -clicks, Border-select, and so on.

One word about the Bézier control points: when you select the main central vertex, the two handles are automatically selected too, so you can grab it as a whole, without creating an angle in the curve. However, when you select a handle, only this vertex is selected, allowing you to modify this control vector…

L (or CtrlL) will add to the selection the cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same curve. Note that for Bézier, using L with a handle selected will select the whole control point and all the linked ones.

Select Menu

With curves, all "advanced" selection options are regrouped in the Select menu of the 3D views header. Let's detail them.

Random...
Inverse
Select/Deselect All
Border Select
	All these options have the same meaning and behavior as in Object mode (and the specifics of Border Select in Edit mode have already been discussed here).

Every Nth

Mode: Edit mode

Hotkey: None

Menu: Select » Every Nth

This only works if you already have at least one control point selected. Using the current selection, it will add to it every nth control point, before and after the initial selection. The "selection step" is specified in the N pop-up numeric field shown during the tool start.

Select/Deselect First/Last

Mode: Edit mode

Hotkey: None

Menu: Select » Select/Deselect First, Select » Select/Deselect Last

These commands will toggle the selection of the first or last control point(s) of the curve(s) in the object. This is useful to quickly find the start of a curve (e.g. when using it as path…).

Select Next/Previous

Mode: Edit mode

Hotkey: None

Menu: Select » Select Next, Select » Select Previous

These commands will select the next or previous control point(s), based on the current selection (i.e. the control points following or preceding the selected ones along the curve).

More and Less

Mode: Edit mode

Hotkey: Ctrl+ NumPad/Ctrl- NumPad

Menu: Select » More/Less

These two options are complementary and similar to those for meshes. Their purpose, based on the currently selected control points, is to reduce or enlarge this selection.

The algorithm is the same as with meshes, but results are more easy to understand:

- More: for each selected control point, select **all** its linked points (i.e. one or two…).
- Less: for each selected control point, if **all** points linked to this point are selected, keep this one selected. Otherwise, de-select it.

This implies two points:

- First, when **all** control points of a curve are selected, nothing will happen (as for Less, all linked points are always selected, and of course, More can't add any). Conversely, the same goes when no control points are selected.
- Second, these tools will never "go outside" of a curve (they will never "jump" to another curve in the same object).

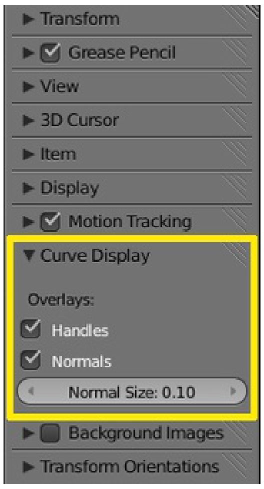
Curve Editing

This page covers the basics of curve editing. Curve basics, selecting and advanced editing are covered in the following pages:

- Curve basics
- Curve Selecting
- Advanced Curve Editing

Curve Display

Display Options



Curve Display panel

When in Edit mode, the Properties Shelf (N) contains options in the Curve Display panel for how curves are displayed in the 3D viewport.

Handles
: Toggles the display of Bezier handles while in edit mode. This does not affect the appearance of the curve itself.

Normals
: Toggles the display of Curve Normals.

Normal Size
: Sets the display scale of curve normals.

Hiding Elements

When in Edit mode, you can hide and reveal elements from the display. This can be useful in complex models with many elements on the Screen.

Hide Selected elements
: Use H, or the Curve » Show/Hide » Hide Selected menu option from the 3D window header.

Show Hidden elements
: Use AltH, or the Curve » Show/Hide » Show Hidden menu option from the 3D window header.

Hide Unselected elements
: Use ⇧ ShiftH, or the Curve » Show/Hide » Hide Unselected menu option from the 3D window header.

Basic Curve Editing (translation, rotation, scale)

Mode: Edit mode

Hotkey: G/R/S

Menu: Curve » Transform » Grab/Move, Rotate, Scale, …

Like other elements in Blender, Curve control points can be grabbed/moved (G), rotated (R) or scaled (S) as described in the Basic Transformations section. When in Edit mode, proportional editing is also available for transformation actions.

Snapping

Mode: Edit mode

Panel: Curve Tools (Editing context)

Mesh snapping also works with curve components. Both control points and their handles will be affected by snapping, except for within itself (other components of the active curve). Snapping works with 2D curves but points will be constrained to the local XY axes.

Deforming Tools

Mode: Edit mode

Menu: Curve » Transform

The To Sphere, Shear, Wrap and Push/Pull transform tools are described in the Transformations sections. The two other tools, Tilt and Shrink/Fatten Radius are related to Curve Extrusion.

Smoothing

Mode: Edit mode

Hotkey: W » smooth

Curve smoothing is available through the specials menu. For Bézier curves, this smoothing operation currently only smooths the positions of control points and not their tangents. End points are also constrained when smoothing.

Mirror

Mode: Edit mode

Hotkey: CtrlM

Menu: Curve » Mirror

The Mirror tool is also available, behaving exactly as with mesh vertices,

---

Set Bézier Handle Type

Mode: Edit mode

Panel: Curve Tools » Handles

Hotkey: V

Menu: Curve » Control Points » Set Handle Type

Handle types are a property of Bézier curves. and can be used to alter features of the curve. For example, switching to Vector handles can be used to create curves with sharp corners. Read the Bézier curves page for more details.

Extending Curves

Mode: Edit mode

Hotkey: Ctrl LMB or E

Menu: Curve » Extrude

Once a curve is created you can add new segments (in fact, new control points defining new segments), either by extruding, or placing new handles with Ctrl LMB clicks. Each new segment is added to one end of the curve. A new segment will only be added if a single vertex, or handle, at one end of the curve is selected. If two or more control points are selected, a new Bézier closed curve is started.

Unlike mesh editing, you cannot create a new curve inside the edited object by Ctrl LMB -clicking without any control points selected. to do so, you can cut an existing curve in two parts (by deleting a segment), copying an existing one (⇧ ShiftD), or add a new one through the menu.

Subdivision

Mode: Edit mode

Panel: Curve Tools (Editing context)

Hotkey: W

Menu: Curve » Segments » Subdivide

Curve subdivision simply subdivides all selected segments by adding one or more control points between the selected segments. To control the number of cuts, press W to make a single subdivision. Then press F6 to bring up the Number of Cuts menu.

Duplication

Mode: Edit mode

Hotkey: ⇧ ShiftD

Menu: Curve » Duplicate

This command duplicates the selected control points, along with the curve segments implicitly selected (if any). The copy is selected and placed in Grab mode, so you can move it to another place.

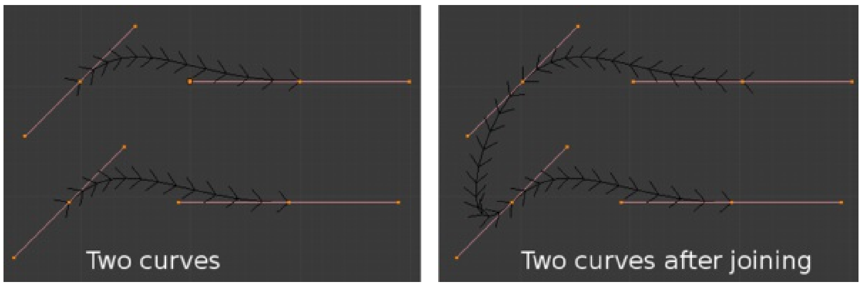
Joining Curve Segments

Mode: Edit mode

Hotkey: F

Menu: Curve » Make Segment

Two open curves can be combined into one by creating a segment between the two curves. To join two separated curves, select one end control point from each curve then press F. The two curves are joined by a segment to become a single curve.



Curves before and after joining

Additionally, you can close a curve by joining the endpoints but note that you can only join curves of the same type (i.e. Bézier with Bézier, NURBS with NURBS)

Separating Curves

Mode: Edit mode

Hotkey: P

Menu: Curve » Separate

Curve objects that are made of multiple distinct curves can be separated into their own objects by selecting the desired segments and pressing P. Note, if there is only one curve in a Curve object, pressing P will create a new Curve object with no control points.

Deleting Elements

Mode: Edit mode

Hotkey: X or Del

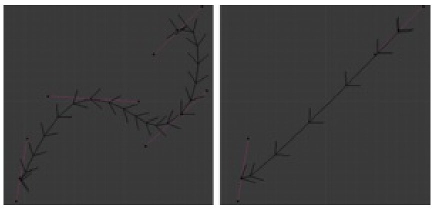
Menu: Curve » Delete...

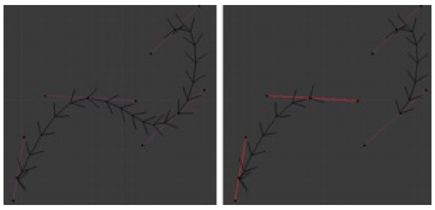
The Erase pop-up menu of curves offers you three options:

Selected
: This will delete the selected control points, *without* breaking the curve (i.e. the adjacent points will be directly linked, joined, once the intermediary ones are deleted). Remember that NURBS order cannot be higher than its number of control points, so it might decrease when you delete some control point. Of course, when only one point remains, there is no more visible curve, and when all points are deleted, the curve itself is deleted.

Segment
: This option is somewhat the opposite to the preceding one, as it will cut the curve, without removing any control points, by erasing one selected segment.
This option always removes *only one segment* (the last “selected” one), even when several are in the selection. So to delete all segments in your selection, you’ll have to repetitively use the same erase option…

All
: As with meshes, this deletes everything in the object!


Deleting Curve Selected


Deleting Curve segments

Opening and Closing a Curve

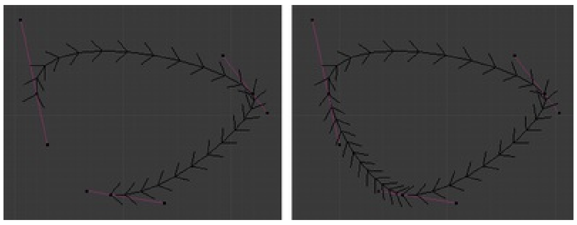
Mode: Edit mode

Hotkey: AltC

Menu: Curve » Toggle Cyclic

This toggles between an open curve and closed curve (Cyclic). Only curves with at least one selected control point will be closed/open. The shape of the closing segment is based on the start and end handles for Bézier curves, and as usual on adjacent control points for NURBS. The only time a handle is adjusted after closing is if the handle is an Auto one. (*Open curve*) and (*Closed curve*) is the same Bézier curve open and closed.

This action only works on the original starting control-point or the last control-point added. Deleting a segment(s) doesn't change how the action applies; it still operates only on the starting and last control-points. This means that AltC may actually join two curves instead of closing a single curve! Remember that when a 2D curve is closed, it creates a renderable flat face.


Open and Closed curves.

Switch Direction

Mode: Edit mode

Hotkey: W » 2 NumPad

Menu: Curve » Segments » Switch Direction, Specials » Switch Direction

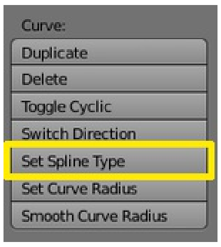
This command will "reverse" the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). This is mainly useful when using a curve as path, or using the bevel and taper options.

Converting Tools

Converting Curve Type

Mode: Edit mode

Panel: Curve Tools»Set Spline type



Set Spline Type button

You can convert splines in a curve object between Bézier, NURBS, and Poly curves. Press T to bring up the Toolshelf. Clicking on the Set Spline Type button will allow you to select the Spline type (Poly, Bézier or NURBS).

Note, this is not a "smart" conversion, i.e. Blender does not try to keep the same shape, nor the same number of control points. For example, when converting a NURBS to a Bézier, each group of three NURBS control points become a unique Bézier one (center point and two handles).

Convert Curve to Mesh

Mode: Object mode

Hotkey: AltC

Menu: Object » Convert to

There is also an "external" conversion, from curve to mesh, that only works in Object mode. It transforms a Curve object in a Mesh one, using the curve resolution to create edges and vertices. Note that it also keeps the faces and volumes created by closed and extruded curves.

Convert Mesh to Curve

Mode: Object mode

Hotkey: AltC

Menu: Object » Convert to

Mesh objects that consist of a series of connected vertices can be converted into curve objects. The resulting curve will be a Poly curve type, but can be converted to have smooth segments as described above.

Curve Parenting

Mode: Edit mode

Hotkey: CtrlP

You can make other selected objects children of one or three control points CtrlP, as with mesh objects.

Select either 1 or 3 control points, then Ctrl RMB another object and use CtrlP to make a vertex parent.

Hooks

Mode: Edit mode

Hotkey: CtrlH

Menu: Curve » control points » hooks

Hooks can be added to control one or more points with other objects.

Set Goal Weight

Mode: Edit mode

Menu: W » Set Goal Weight

Set Goal Weight
: This sets the “goal weight” of selected control points, which is used when a curve has Soft Body physics, forcing the curve to "stick" to their original positions, based on the weight.

Curve Deform

Curve Deform provides a simple but efficient method of defining a deformation on a mesh. By parenting a mesh object to a curve, you can deform the mesh up or down the curve by moving the mesh along, or orthogonal to, the dominant axis. This is a most useful tool to make an object follow a complex path, like e.g. a sheet of paper inside a printer, a film inside a camera, the water of a canal…

The Curve Deform works on a (global) dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction, the mesh will traverse along the curve. Moving the mesh in an orthogonal direction will move the mesh object closer or further away from the curve. The default settings in Blender map the Y axis to the dominant axis. When you move the object beyond the curve endings the object will continue to deform based on the direction vector of the curve endings.

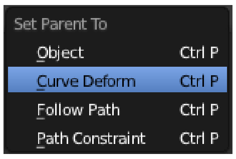
If the “curve path” is 3D, the Tilt value of its control points will be used (see the Extrusion section above) to twist the “curved” object around it. Unfortunately, the other Radius property is not used (it would have been possible, for example, to make it control the size of the “curved” object…).

A Tip

Try to position your object over the curve immediately after you have added it, before adding the curve deform. This gives the best control over how the deformation works.

Use modifiers!
The Curve Deform relationship is now also a modifier, called Curve. The Curve modifier function acts the same as its counterpart, except that when the modifier is used, the “dominant axis” is set inside its properties – and the Track X/Y/Z buttons no longer have an effect on it. And you have some goodies, like the possibility, if “curving” a mesh, to only curve one of its vertex groups…

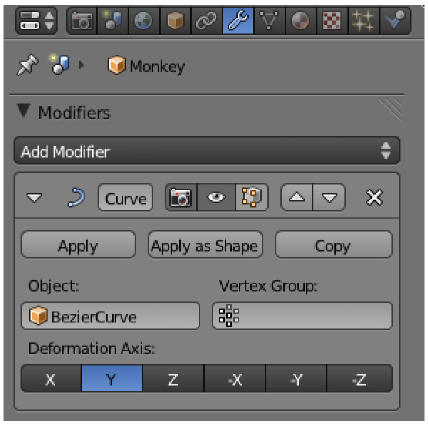
Interface



Make Parent menu.

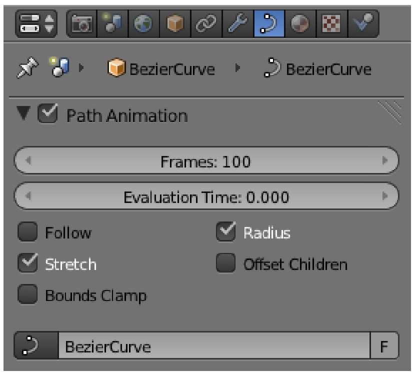
When parenting an object (mesh, curve, meta, …) to a curve (CtrlP), you will be presented with a menu (*Make Parent menu*).

By selecting Curve Deform, you enable the curve deform function on the mesh object.



Anim settings panel.

The dominant axis setting is set on the mesh object. By default the dominant axis in Blender is Y. This can be changed by selecting one of the Track X, Y or Z buttons in the Anim Panel, (*Anim settings panel*), in Object context (F7).



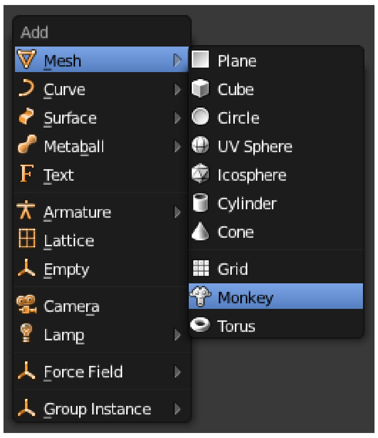
Curve and Surface panel.

Cyclic (or closed) curves work as expected where the object deformations traverse along the path in cycles. Note however that when you have more than one curve in the “parent” object, its “children” will only follow the first one.

The Stretch curve option allows you to let the mesh object stretch, or squeeze, over the entire curve. This option is in Editing Context (F9), for the “parent” curve. See (*Curve and Surface panel*).

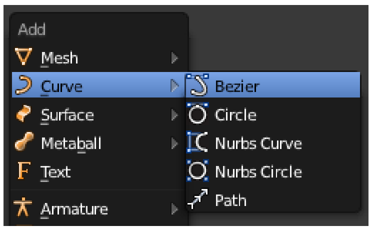
Example

Let’s make a simple example:



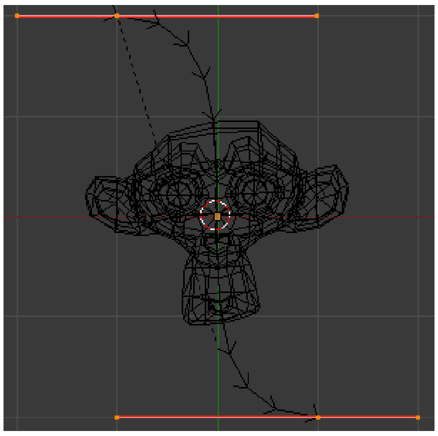
Add a Monkey!

- Remove default cube object from scene and add a Monkey (⇧ ShiftA » Add » Mesh » Monkey, *Add a Monkey!*)!
- Press ⇆ Tab to exit Edit mode.

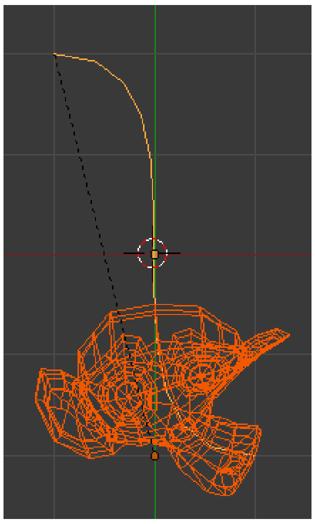


Add a Curve.

- Now add a curve (⇧ ShiftA » Add » Curve » Bezier Curve, *Add a Curve*).


Edit Curve.

- While in Edit mode, move the control points of the curve as shown in (*Edit Curve*), then exit Edit mode (⇆ Tab).


Monkey on a Curve.

- Now, you can use the new, modern, modifier way of “curving” the Monkey:
  - Select the Monkey (RMB).
  - In the Editing context (F9), Modifiers panel, add a Curve modifier.
  - Type the name of the curve (should be “`Curve`”) in the Ob field of the modifier, and optionally change the dominant axis to Y.
- Or you can choose the old, deprecated method (note that it creates a “virtual” modifier…):
  - Select the Monkey (RMB), and then shift select the curve (⇧ Shift RMB).
  - Press CtrlP to open up the Make Parent menu.
  - Select Curve Deform (*Make Parent menu*).
- The Monkey should be positioned on the curve, as in (*Monkey on a Curve*).
- Now if you select the Monkey (RMB), and move it (G), in the Y-direction (the dominant axis by default), the monkey will deform nicely along the curve.

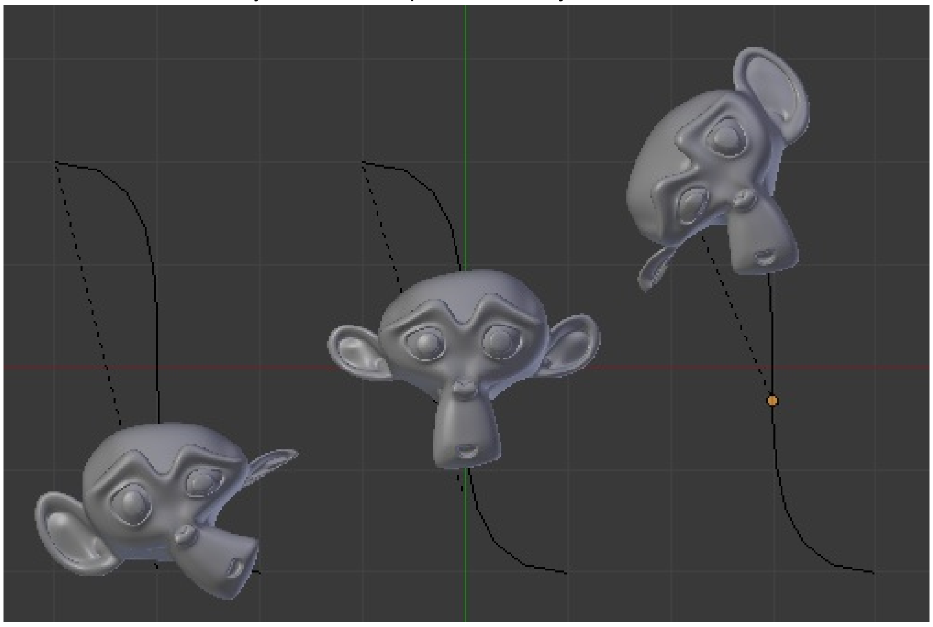
A Tip

If you press MMB (or one of the X/Y/Z keys) while moving the Monkey you will constrain the movement to one axis only.

- In (*Monkey deformations*), you can see the Monkey at different positions along the curve. To get a cleaner view over the deformation I have activated SubSurf with Subdiv to **2**, and Set Smooth on the Monkey mesh (F9 to get Editing context).

A Tip

Moving the Monkey in directions other than the dominant axis will create some odd deformations. Sometimes this is what you want to achieve, so you’ll need to experiment and try it out!


Monkey deformations.

Curve Extrusion

This section covers methods for extruding curves, or giving them thickness, and how to control the thickness along the path.

Extrusion

Mode: Object or Edit mode

Panel: Curve and Surface (Editing context, F9)

Ah! The extrusion! Probably the most interesting tool of the curves for modeling, especially with the bevel/taper/Tilt/Radius options… Note that this has nothing to do with the Extrude (E) command, described in the previous page!

We will see the different settings, depending on their scope of action:

Width
: This controls the position of the extruded "border" of the curve, relative to the curve itself. With closed 2D curves (see below), it is quite simple to understand – with a Width greater than **1.0**, the extruded volume is wider, with a Width of **1.0**, the border tightly follows the curve, and with a Width lower than **1.0**, the volume is narrower… The same principle remains for open 2D and 3D curves, but the way the "outside" and "inside" of the curve is determined seems a bit odd…
It has the same effect with extruded "bevel" objects…

Tilt
: This setting – unfortunately, you can never see its value anywhere in Blender – controls the "twisting angle" around the curve for each point – so it is only relevant with 3D curves!
You set it using the Tilt transform tool (T, or Curve » Transform » Tilt), and you can reset it to its default value (i.e. perpendicular to the original curve plane) with AltT (or Curve » Control Points » Clear Tilt).
With NURBS, the tilt is always smoothly interpolated. However, with Bézier, you can choose the interpolation algorithm to use in the Tilt Interpolation drop-down list of the Curve Tools panel (you will find the classical Linear, Cardinal, B Spline and Ease options…).

Simple Extrusion

Let's first see the "simple" extrusion of curves, without additional bevel/taper objects.

Extrude
: This controls the width (or height) of the extrusion. The real size is of course dependent on the scale of the underlying object, but with a scale of one, an Extrusion of **1.0** will extrude the curve one BU in both directions, along the axis perpendicular to the curve's plane (see below for specifics of 3D curves…).
If set to **0.0**, there is no "simple" extrusion!

Bevel Depth
: This will add a bevel to the extrusion. See below for its effects…
Note that the bevel makes the extrusion wider and higher.
If set to **0.0**, there is no bevel (max value: **2.0**).

Bev Resol
: Controls the resolution of the bevel created by a Bevel Depth higher than zero.
If set the **0** (the default), the bevel is a simple "flat" surface.
Higher values will smooth, round off the bevel, similar to the resolution settings of the curve itself…

We have three sub-classes of results, depending on whether the curve is open or closed or 3D:

Open 2D Curve
: The extrusion will create a "wall" or "ribbon" following the curve shape. If using a Bevel Depth, the wall becomes a sort of slide or gutter. Note the direction of this bevel is sometimes strange and unpredictable, often the reverse of what you would get with the same curve closed… You can inverse this direction by switching the direction of the curve.
This allows you, e.g., to quickly simulate a marble rolling down a complex slide, by combining an extruded beveled curve, and a sphere with a Follow Path constraint set against this curve…

Closed 2D Curve
: This is probably the most useful situation, as it will quickly create a volume, with (by default) two flat and parallel surfaces filling the two sides of the extruded "wall". You can remove one or both of these faces by disabling the Back and/or Front toggle buttons next to the 3D one.
The optional bevel will always be "right-oriented" here, allowing you to smooth out the "edges" of the volume.

3D Curve
: Here the fact that the curve is closed or not has no importance – you will never get a volume with an extruded 3D curve, only a wall or ribbon, like with open 2D curves.
However, there is one more feature with 3D curves: the Tilt of the control points (see above). It will make the ribbon twist around the curve … to create a Möbius strip, for example!

Advanced Extrusion

These extrusions use one or two additional curve objects, to create very complex organic shapes.

To enable this type of extrusion, you have to type a valid curve object name in the BevOb field of the curve you are going to use as the "spinal column" of your extrusion. The "bevel" curve will control the cross section of the extruded object. Whether the BevOb curve is 2D or 3D has no importance, but if it is closed, it will create a "tube-like" extrusion; otherwise you will get a sort of gutter or slide object…

The object is extruded along the whole length of all internal curves. By default, the width of the extrusion is constant, but you have two ways to control it, the Radius property of control points, and the “taper” object.

The Radius of the points is set using the Shrink/Fatten Radius transform tool (AltS, or Curve » Transform » Shrink/Fatten Radius), or with the Set Radius entry in the Specials menu (W). Here again, you unfortunately cannot visualize anywhere the Radius of a given control point…

The Radius allows you to directly control the width of the extrusion along the “spinal” curve. As for Tilt (see above), you can choose the interpolation algorithm used for Bézier curves, in the Radius Interpolation drop-down list of the Curve Tools panel.

But you have another, more precise option: the “taper” object. As for the “bevel” one, you set its name in the TaperOb field of the main curve – it must be an *open curve*. The taper curve is evaluated along *the local X axis*, using *the local Y axis* for width control. Note also that:

- The taper is applied independently to all curves of the extruded object.
- Only the first curve in a TaperOb is evaluated, even if you have several separated segments.
- The scaling starts at the first control-point on the left and moves along the curve to the last control-point on the right.
- Negative scaling, (negative local Y on the taper curve) is possible as well. However, rendering artifacts may appear.
- It scales the width of normal extrusions based on evaluating the taper curve, which means sharp corners on the taper curve will not be easily visible. You’ll have to heavily level up the resolution (DefResolU) of the base curve.
- With closed curves, the taper curve in TaperOb acts along the whole curve (perimeter of the object), not just the length of the object, and varies the extrusion depth. In these cases, you want the relative height of the TaperOb Taper curve at both ends to be the same, so that the cyclic point (the place where the endpoint of the curve connects to the beginning) is a smooth transition.

Last but not least, with 3D “spinal” curves, the Tilt of the control points can control the twisting of the extruded “bevel” along the curve!

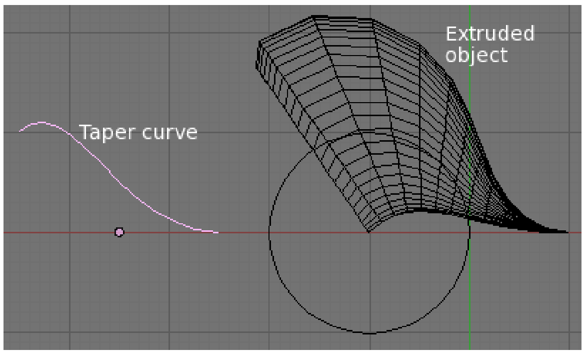
Examples

TODO: add some “simple” extrusion examples.

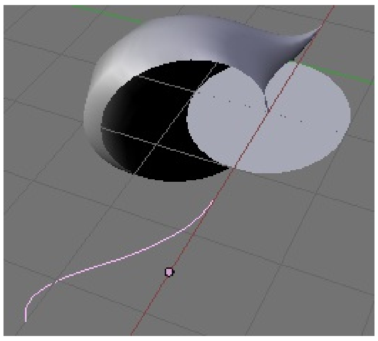
TODO: add some “bevel” extrusion with Radius examples.

Let’s taper a simple curve circle extruded object using a taper curve. Add a curve, then exit Edit mode. Add another one (a closed one, like a circle); call it “`BevelCurve`”, and enter its name in the BevOb field of the first curve (Editing context F9, Curve and Surface panel). We now have a pipe. Add a third curve while in Object mode and call it “`TaperCurve`”. Adjust the left control-point by raising it up about 5 units.

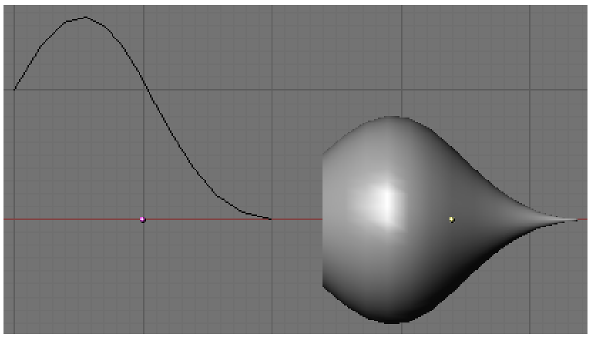
Now return to the Editing context, and edit the first curve’s TaperOb field in Curve and Surface panel to reference the new taper curve which we called “`TaperCurve`”. When you hit enter the taper curve is applied immediately, with the results shown in (*Taper extruded curve*).



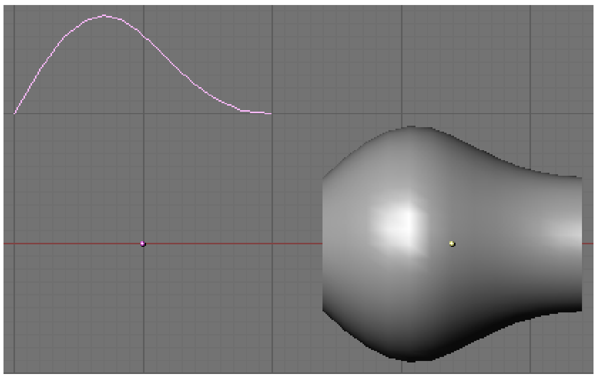
Taper extruded curve.


Taper solid mode.

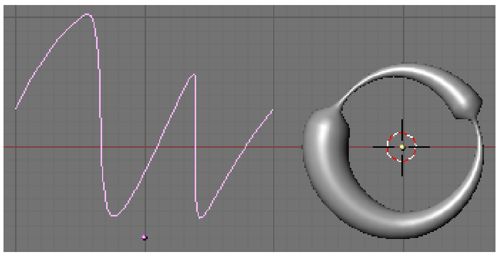
You can see the **taper curve** being applied to the **extruded object**. Notice how the pipe’s volume shrinks to nothing as the taper curve goes from left to right. If the taper curve went below the local Y axis the pipe’s inside would become the outside, which would lead to rendering artifacts. Of course as an artist that may be what you are looking for!


Taper example 1.

In (*Taper example 1*) you can clearly see the effect the left taper curve has on the right curve object. Here the left taper curve is closer to the object center and that results in a smaller curve object to the right.


Taper example 2.

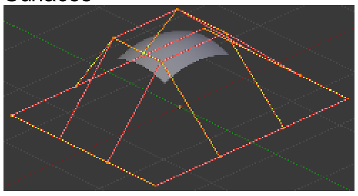
In (*Taper example 2*) a control point in the taper curve to the left is moved away from the center and that gives a wider result to the curve object on the right.


Taper example 3.

In (*Taper example 3*), we see the use of a more irregular taper curve applied to a curve circle.

TODO: add some “bevel” extrusion with Tilt examples.

Surfaces


Surface.

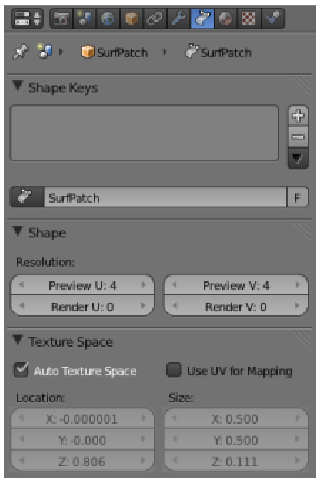
Curves are 2D objects, and surfaces are their 3D extension. Note however that in Blender, you only have NURBS surfaces, no Bézier (you have the Bezier knot type, though; see below), nor polygonal (but for these, you have meshes!). Even though curves and surfaces share the same object type (with texts also…), they are not the same thing; for example, you cannot have in the same object both curves and surfaces.

As surfaces are 2D, they have two interpolation axes, U (as for curves) and V. It is important to understand that *you can control the interpolation rules (knot, order, resolution) independently for each of these two dimensions* (the U and V fields for all these settings, of course).

You may ask yourself "but the surface appears to be 3D, why is it only 2D?". In order to be 3D, the object needs to have "Volume," and a surface, even when it is closed, doesn't have volume; it is infinitely thin. If it had a volume the surface would have a thickness (its third dimension). Hence, it's only a 2D object, and has only two interpolation dimensions or axes or coordinates (if you know a bit of math, think of non-euclidean geometry – well, surfaces are just non-euclidean 2D planes…). To take a more "real life" example, you can roll a sheet of paper to create a cylinder; well, even if it "draws" a volume, the sheet itself will remain a (nearly…) 2D object!

In fact, surfaces are very similar to the results you get when extruding a curve (by the way, I think it should be possible to convert an extruded curve to a surface, at least when only made of NURBS – but Blender cannot do it currently…).

Finding Surface Tools


SurfPatch › SurfPatch

▼ Shape Keys

SurfPatch F

▼ Shape

Resolution:

Preview U: 4 | Preview V: 4

Render U: 0 | Render V: 0

▼ Texture Space

☑ Auto Texture Space ☐ Use UV for Mapping

Location: X: -0.000001 Y: -0.000 Z: 0.806

Size: X: 0.500 Y: 0.500 Z: 0.111

Surface Tools.

The panels of the Editing context are the same as for curves, just with fewer options… And as usual, you have the Select and Surface menus in the 3D view headers, and the Specials (W) pop-up one.

Visualization

There is nearly no difference from NURBS curves, except that the `U` direction is indicated by yellow grid lines, and the `V` one is materialized by pink grid lines, as you can see in (*Surface*).

You can hide and reveal control points just as with curves, and you have the same draw options in the Curve Tools panel.

Surface Structure

Many of the concepts from curves, especially NURBS ones, carry directly over to NURBS surfaces, such as control points, Order, Weight, Resolution, etc. Here we will just talk about the differences.

It is very important to understand the difference between NURBS curves and NURBS surfaces: the first one has one dimension, the latter has two. Blender internally treats NURBS surfaces and NURBS curves completely differently. There are several attributes that separate them but the most important is that a NURBS curve has a single interpolation axis (U) and a NURBS surface has two interpolation axes (U and V).

However, you can have "2D" surfaces made of curves (using the extrusion tools, or, to a lesser extent, the filling of closed 2D curves. And you can have "1D" curves made of surfaces, like a NURBS surface with only one row (either in U or V direction) of control points produces only a curve…

Visually you can tell which is which by entering Edit mode and looking at the 3D window's header: either the header shows "Surface" or "Curve" as one of the menu choices. Also, you can extrude a whole NURBS surface curve to create a surface, but you can't with a simple NURBS curve (we talk here about the "standard" Extrude tool, the one activated with the E shortcut, not the quite-specific curve extrusion tools – yes, I know, it's not easy to follow…).

Control Points, Rows and Grid

Control points for NURBS surfaces are the same as for NURBS curves. However, their layout is quite constraining. The concept of "segment" disappears, replaced by "rows" and the overall "grid".

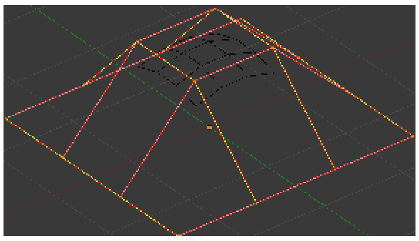
A "row" is a set of control points forming one "line" in one interpolation direction (a bit similar to edge loops for meshes). So you have "U-rows" and "V-rows" in a NURBS surface. The key point is that *all rows of a given type (U or V) have the same number of control points*. Each control point belongs to exactly one U-row and one V-row.

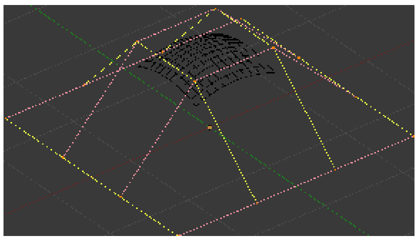
All this forms a "grid", or "cage", the shape of which controls the shape of the NURBS surface. A bit like a lattice…

This is very important to grasp: you cannot add a single control point to a NURBS surface; you have to add a whole U- or V-row at once (in practice, you will usually use the Extrude tool, or perhaps the Duplicate one, to add those…), containing exactly the same number of points as the others. This also means that you will only be able to "merge" different pieces of surfaces if at least one of their rows match together.

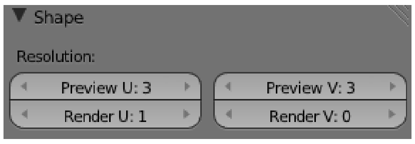
Surface Resolution

Just like NURBS curves, Resolution controls the detail of the surface. The higher the Resolution the more detailed and smoother the surface is. The lower the Resolution the rougher the surface. However, here you have two resolution settings, one for each interpolation axis (U and V). Note that unlike with curves, you have only one resolution (the Resol U and V fields, in the Curve Tools panel)…


Resolution 1x1.


Resolution 3x3.

(*Resolution 1x1*) is an example of a surface resolution of 3 for both `U` and `V`. (*Resolution 3x3 surface*) is an example of a surface resolution of 12 for both `U` and `V`.



Preview U: 3
Render U: 1
Preview V: 3
Render V: 0

Resolution panel.

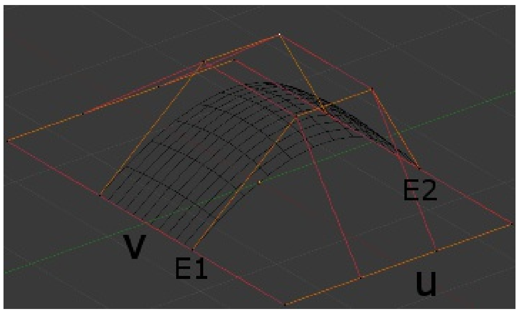
You can adjust the resolution separately for both preview and render, to not slow things down in the viewport, but still get good render results.

Closed and Open Surfaces

Like curves, surfaces can be closed (cyclical) or open, independently in both directions, allowing you to easily create a tube, donut or sphere shape, and they can be drawn as "solids" in Edit mode. This makes working with surfaces quite easy.

Knots

Just like with NURBS curves, NURBS surfaces have two knot vectors, one for each U and V axis. Here again, they can be one of Uniform, Endpoint, or Bezier, with the same properties as for curves. And as with curves, only open surfaces (in the relevant direction) are affected by this setting…



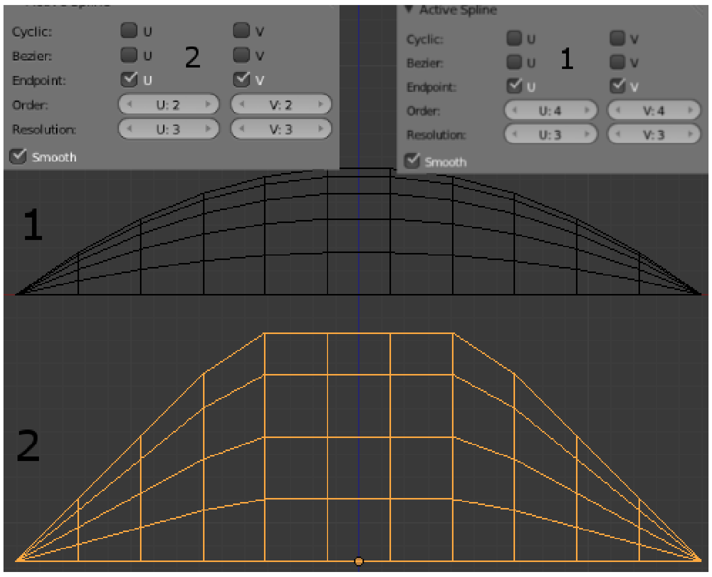
Endpoint U.

In (*Endpoint U*), the U interpolation axis is labeled as "`U`" and the V interpolation axis is labeled as "`V`". The `U`'s interpolation axis has been set to Endpoint and as such the surface now extends to the outer edges from "`E1`" to "`E2`" along the `U` interpolation axis.

To cause the surface to extend to all edges you would set the `V`'s axis to Endpoint as well.

Order

One more time, this property is the same as with NURBS Curves; it specifies how much the control points are taken into account for calculating the curve of the surface shape. For high Orders, (*1*), the surface pulls away from the control points, creating a smoother surface – assuming that the resolution is high enough. For lowest Orders, (*2*), the surface follows the control points, creating a surface that tends to follow the grid cage.


Cyclic: ☐ U ☐ V
Bezier: ☐ U ☐ V
Endpoint: ✓ U ✓ V
Order: U: 2 V: 2
Resolution: U: 3 V: 3
✓ Smooth

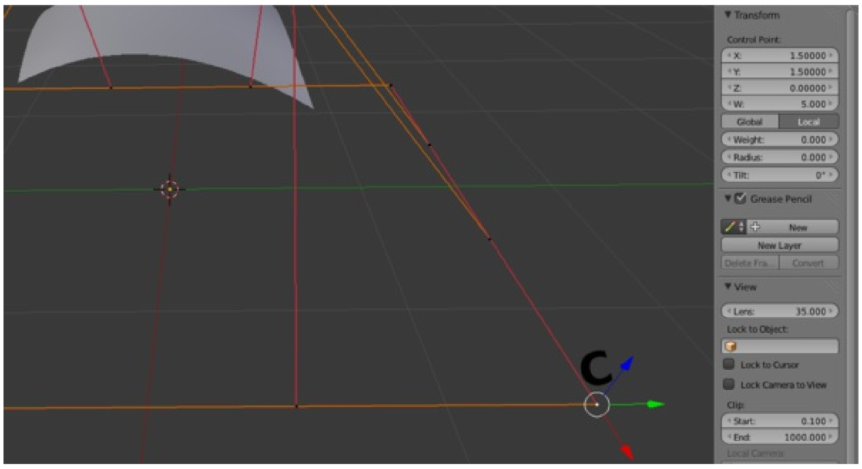
Active Spline
Cyclic: ☐ U ☐ V
Bezier: ☐ U ☐ V
Endpoint: ✓ U ✓ V
Order: U: 4 V: 4
Resolution: U: 3 V: 3
✓ Smooth

Order 2 and order 4 surface.

For illustration purposes, in both (*Order 4 surface*) and (*Order 2 surface*), the knot vectors were set to Endpoint, causing the surface to extend to all edges.

You can set independently the order for each interpolation axis, and like curves, it cannot be lower than **2**, and higher than **6** or the number of control points on the relevant axis.

Weight


Transform
Control Point:
X: 1.50000
Y: 1.50000
Z: 0.00000
W: 5.000
Global Local
Weight: 0.000
Radius: 0.000
Tilt: 0°
✓ Grease Pencil
New
New Layer
Delete Fra... Convert
View
Lens: 35.000
Lock to Object:
☐ Lock to Cursor
☐ Lock Camera to View
Clip:
Start: 0.100
End: 1000.000
Local Camera

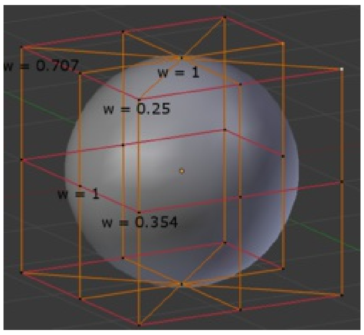
Guess what? Yes, it works exactly like NURBS Curves! Weight specifies how much each control point "pulls" on the curve.

In (*Surface Weight 5*), a single control point, labeled "`C`", has had its Weight set to **5.0** while all others are at their default of **1.0**. As you can see, that control point *pulls* the surface towards it.

If all the control points have the same Weight then each effectively cancels each other out. It is the difference in the weights that cause the surface to move towards or away from a control point.

The Weight of any particular control point is visible in the Transform Properties panel (N), *in the W field* (and not the Weight field…).

Preset Weights



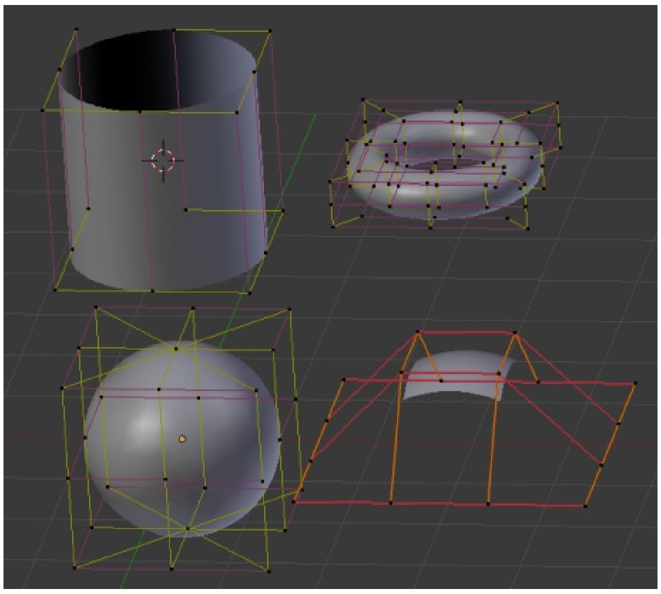
A sphere surface.

NURBS can create pure shapes such as circles, cylinders, and spheres (note that a Bézier circle is not a pure circle). To create pure circles, globes, or cylinders, you must set to specific values the weights of the control points – some of which are provided as presets in the Curve Tools panel (lower right corner). This is not intuitive, and you should read more on NURBS before trying this.

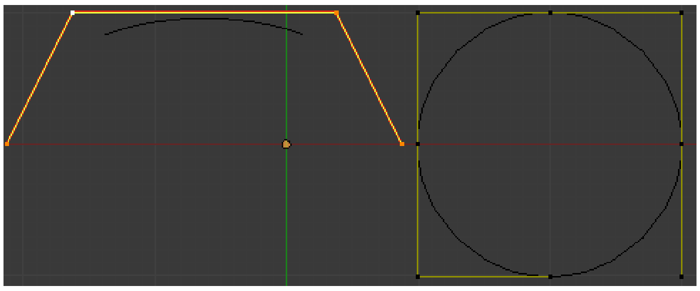
We saw with 1D NURBS curves how to create a circle; let's see how to create a sphere with 2D surfaces. It is the same principle – you'll note that the four different weights needed for creating a sphere (**1.0**, **0.707** = `sqrt(0.5)`, **0.354** = `sqrt(2)/4`, and **0.25**) are the four presets available in the Curve Tools panel…

Primitives

To help get started in creating surfaces there are four preset NURBS surfaces, found in the Add » Surface menu: NURBS Surface, NURBS Tube, NURBS Sphere and NURBS Torus.


NURBS surface primitives.

There are also two preset NURBS surface curves (with only one control point on each V-row): NURBS Curve and NURBS Circle.


NURBS curve primitives.

Note how a circle NURBS surface is never filled, unlike its "real" curve counterpart…

Surface Selection

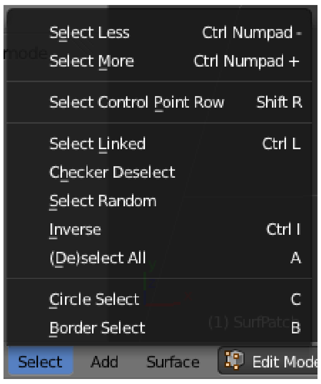
Surface selection in Edit mode is very similar to NURBS curve selection. The basic tools are the same as with meshes, so you can select a simple control point with a LMB -click, add to current selection with ⇧ Shift LMB -clicks, Border-select, and so on.

L (or CtrlL) will add to the selection the mouse cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same surface.

Select Menu

The Select menu (3D view headers) is even simpler than for curves…

> All these options have the same meaning and behavior as in Object mode (and the specificities of Border Select in Edit mode have already been discussed here).



Every Nth

Mode: Edit mode

Hotkey: None

Menu: Select » Every Nth

This is the same option as for curve selection. However, the behavior of the N ("selection step") parameter in the 2D of a NURBS surface "cage" seems quite difficult to understand…

Control Point Row

Mode: Edit mode

Hotkey: ⇧ ShiftR

Menu: Select » Control Point Row

This option works a bit like edge loop selection for meshes, inasmuch it selects a whole row of control points, based on the active (the last selected) one. The first time you hit ⇧ ShiftR, the V-row passing through (containing) the active point will be *added to the current selection*. If you use again this shortcut, you will toggle between the U- and V-row of this point, *removing everything else from the selection*.

More and Less

Mode: Edit mode

Hotkey: Ctrl+ NumPad/Ctrl- NumPad

Menu: Select » More/Less

These two options are complementary and very similar to those for meshes. Their purpose, based on current selected control points, is to reduce or enlarge this selection.

The algorithm is the same as with meshes:

- More: for each selected control point, select **all** its linked points (i.e. two, three or four).
- Less: for each selected control point, if **all** points linked to this point are selected, keep it selected. For all other selected control points, de-select them.

This implies two points:

- First, when **all** control points of a surface are selected, nothing will happen (as for Less, all linked points are always selected, and of course, More can't add any). Conversely, the same goes when no control point is selected.
- Second, these tools will never "go outside" of a surface (they will never "jump" to another surface in the same object).

Surface Editing

Surface editing has even fewer tools and options than its curve counterpart – and has many common points with it… So this page covers (or tries to cover) all the subjects, from the basics of surface editing to more advanced topics, like retopology.

Basic Surface Editing (translation, rotation, scale)

Mode: Edit mode

Hotkey: G/R/S

Menu: Surface » Transform » Grab/Move, Rotate, Scale, …

Once you have a selection of one or more control points, you can grab/move (G), rotate (R) or scale (S) them, like many other things in Blender, as described in the Manipulation in 3D Space section.

You also have in Edit mode an extra option when using these basic manipulations: the proportional editing.

Advanced Transform Tools

Mode: Edit mode

Menu: Surface » Transform

The To Sphere, Shear, Wrap and Push/Pull transform tools are described in the Mesh Editing chapter. Surfaces have no specific transform tools.

NURBS Control Points Settings

Mode: Edit mode

Panel: Curve Tools (Editing context, F9), and Transform Properties

We saw in a previous page that NURBS control points have a weight, which is the influence of this point on the surface. You set it either using the big Set Weight button in the Curve Tools panel (after having defined the weight in the numeric field to the right), or by directly typing a value in the W numeric field of the Transform Properties panel.

Adding or Extruding

Mode: Edit mode

Hotkey: E (or Ctrl LMB)

Menu: Surface » Extrude

Unlike meshes or curves, you cannot generally directly add new control points to a surface (with Ctrl LMB clicks), as you can only extend a surface by adding a whole U- or V-row at once. The only exception is when working on a NURBS surface curve, i.e. a surface with only one control point on each U- or V-row. In this special case, all works exactly as with curves.

Most of the time, only extrusion is available. As usual, once the tool is activated the extrusion happens immediately and you are placed into Grab mode, ready to drag the new extruded surface to its destination.

There are two things very important to understand:

- Surfaces are **2D** objects – so you can’t extrude anything *inside* a surface (e.g. “inner” row); it wouldn’t make any sense!
- The control “grid” *must remain “squarish”*, which means that you can only extrude a whole row, not parts of rows here and there…

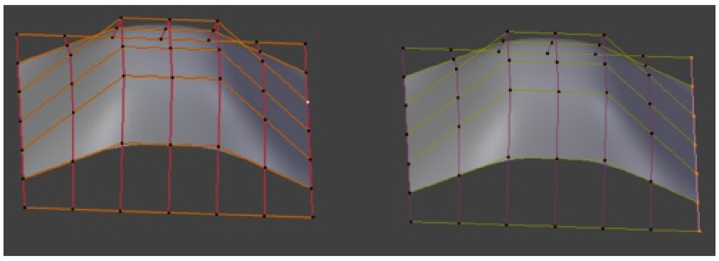
To summarize, the Extrude tool will only work when one and only one whole border row is selected – otherwise nothing happens.

As for curves, you cannot create a new surface in your object out of nowhere, by just Ctrl LMB -clicking with nothing selected. However, unlike for curves, there is no “cut” option allowing you to separate a surface into several parts, so you only can create a new surface by copying an existing one (⇧ ShiftD), or adding a new one (Add menu…).

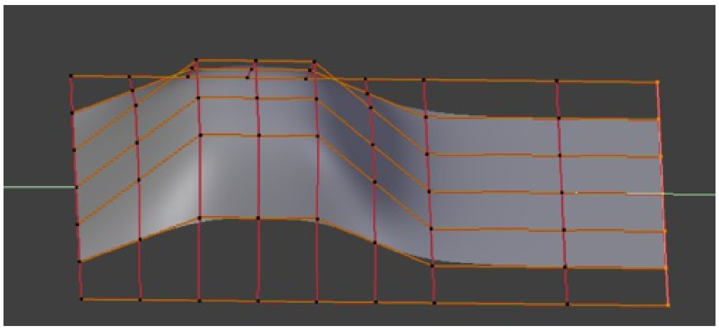
Examples

Images (*Selecting control-point*) to (*Complete*) show a typical extrusion along the side of a surface.

In (*Selecting control-point*) and (⇧ *ShiftR*), a border row of control points were highlighted by selecting a single control point, labeled “C”, and then using the handy row select tool (⇧ ShiftR) to select the rest of the control points.



The edge is then extruded using E as shown in (*Extruding*). Notice how the mesh has bunched up next to the highlighted edge; the area in question is highlighted in a light-gray circular area. That is because the *new*extruded surface section is bunched up there as well.



By moving the new section away from the area, the surface begins to “unbunch”. The direction of movement is marked with a white arrow, labeled “`E`”, and the new section is labeled “`S`”.

You can continue this process of extruding – or adding – new surface sections until you have reached the final shape for your model.

Opening or Closing a Surface

Mode: Edit mode

Hotkey: C

Menu: Surface » Toggle Cyclic

As in curves, surfaces can be closed (cyclic) or open. However, as surfaces are 2D, you can control this property independently along the U and V axes.

To toggle the cyclic property of a surface along one axis, use C and choose either cyclic U or cyclic V from the Toggle pop-up menu. The corresponding surface’s outer edges will join together to form a “closed” surface.

Inner and Outer
Surfaces have an “inner” and “outer” face, the first being black whereas the latter is correctly shaded – there does not seem to be any “double sided” shading option for surfaces…). When you close a surface in one or two directions, you might get an entirely black object! In this case, just switch the “direction” of your surface…

Duplication

Mode: Edit mode

Hotkey: ⇧ ShiftD

Menu: Curve » Duplicate

Well, as with meshes and curves, this command just duplicates the selection. As usual, the copy is selected and placed in Grab mode, so you can move it to another place.

However, with surfaces there are some selections that can’t be duplicated, in which case they will just be placed in Grab mode… In fact, only selections forming *a single valid sub-grid* are copyable; let’s see this in practice:

- You can copy a single control point. From it, you will be able to “extrude” a “surface curve” along the U axis, and then extrude this unique U-row along the V axis to create a real new surface.
- You can copy a single continuous part of a row (or a whole row, of course). This will give you a new **U-row**, even if you selected (part of) a V-row!
- You can copy a single whole sub-grid.

Note that trying to duplicate several valid “sub-grids” (even being single points) at once won’t work; you’ll have to do it one after the other…

Deleting Elements

Mode: Edit mode

Hotkey: X or Del

Menu: Curve » Delete...

The Erase pop-up menu of surfaces offers you two options:

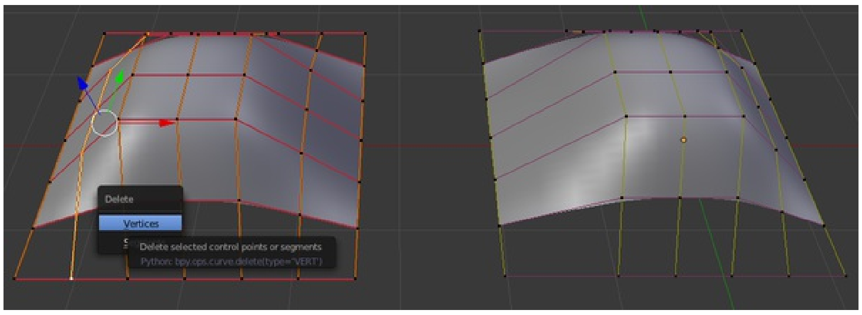
Selected
: This will delete the selected rows, *without* breaking the surface (i.e. the adjacent rows will be directly linked, joined, once the intermediary ones are deleted). The selection must abide by the following rules:

  - Whole rows, and only whole rows must be selected.
  - Only rows along the same axis must be selected (i.e. you can't delete both U- and V-rows at the same time).

  Also remember that NURBS order cannot be higher than its number of control points in a given axis, so it might decrease when you delete some control points… Of course, when only one row remains, the surface becomes a "surface curve"; when only one point remains, there is no more visible surface; and when all points are deleted, the surface itself is deleted.

All
: As with meshes or curves, this deletes everything in the object!

Example


Delete
Vertices
Delete selected control points or segments
Python: bpy.ops.curve.delete(type='VERT')

Before and after

In (*Before*) a row of control points has been selected by initially selecting the control point labeled "`A`" and using ⇧ ShiftR to select the remaining control points. Then, using the Erase menu (X), the *selected* row of control points is erased, resulting in (*After*).

Joining or Merging Surfaces

Mode: Edit mode

Hotkey: F

Menu: Surface » Make Segment

Just like curves, merging two surfaces requires that a single edge, a border row of control points, from two separate surfaces are selected. This means that the surfaces must be part of the same object. For example, you can't join two surfaces while in Object mode – but you can of course, as with any objects of the same type, join two or more Surface objects into one object (CtrlJ) – they just won't be "linked" or merged in a single one… Yes, it's a bit confusing!

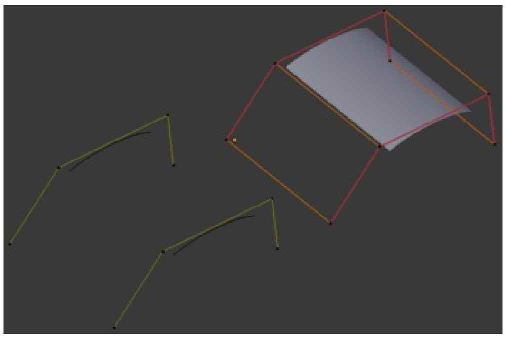
This command is equivalent to creating edges or Faces for meshes (hence its shortcut), and so it only works in Edit mode. The selection must contains only border rows of the same resolution (with the same number of control points), else Blender will try to do its best to guess what to merge with what, or the merge will fail (either silently, or stating that "`Resolution doesn't match`" if rows with different number of points are selected, or that there is "`Too few selections to merge`" if you only selected points in one surface…).

So to avoid problems, you should always only select border rows with the same number of points… Note that you can join a border U-row of one surface with a border V-row of another one, Blender will automatically "invert" the axis of one surface for them to match correctly.

NURBS surface curves are often used to create objects like hulls, as they define cross sections all along the object, and you just have to "skin" them as described above to get a nice, smooth and harmonious shape. See this tutorial for a detailed workflow.

Examples

(*Joining ready*) is an example of two NURBS surface curves, ***not*** NURBS curves, in Edit mode, ready to be joined. (*Joining complete*) is the result of joining the two curves.


Joining ready.

Subdivision

Mode: Edit mode

Panel: Curve Tools1 (Editing context, F9)

Hotkey: W » 1 NumPad

Menu: Surface » Segments » Subdivide, Specials » Subdivide

Surface subdivision is most simple: using either the Subdivide entry in the Specials menu (W), or the Subdivide button of the Curve Tools1 panel, you will subdivide once all *completely selected grids* by subdividing each “quad” into four smaller ones.

If you apply it to a 1D surface (a “surface curve”), this tool works exactly as with curves.

Spin

Mode: Edit mode

Panel: Curve Tools1 (Editing context, F9)

This tool is a bit similar to its mesh counterpart – but with less control and options (in fact, there’s none!).

It only works on selected “surfaces” made of *one U-row*(and not with one V-row), so-called “surface curves”, by “extruding” this “cross section” in a square pattern, automatically adjusting the weights of control points to get a perfect circular extrusion (this also implies closing the surface along the V axis), following exactly the same principle as for the NURBS Tube or NURBS Donut primitives.

Switch Direction

Mode: Edit mode

Hotkey: W » 2 NumPad

Menu: Surface » Segments » Switch Direction, Specials » Switch Direction

This command will “reverse” the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). Mainly useful when using a curve as path, or the bevel and taper options…

Other Specials Options

Mode: Edit mode

Hotkey: W

Menu: Specials

The Specials menu contains exactly the same additional options as for curves – but I suppose Set Radius and Smooth Radius have nothing to do here…

Conversion

As there are only NURBS surfaces, there is no “internal” conversion here.

However, there is an “external” conversion available, from surface to mesh, that only works in Object mode. It transforms a Surface object into a Mesh one, using the surface resolutions in both directions to create faces, edges and vertices.

Retopology

Snapping surface components is the same as is with meshes and curves. See Retopology for more information.

Misc Editing

You have some of the same options as with meshes, or in Object mode. You can separate a given surface (P), make other selected objects children of one or three control points (CtrlP – note however that parenting to three control points has a strange behavior with curves…), or add hooks to control some points with other objects.

The Mirror tool is also available, behaving exactly as with mesh vertices.

Text Objects

Mode: Edit mode (Text)

Panel: Curve and Surface, Font and Char (Editing context, F9)

Hotkey: F9

Menu: Add » Text

Blender font
Agate font
Becker font
Blippo font
Slogan font
This is a text on a curve. driving me in circles

Text Examples.

Text objects are exactly what they sound like: they contain some text. They share the same object type as curves and surfaces, as modern fonts (OpenType, TrueType, etc.) are vectorial, made of curves (generally Béziers).

Blender uses a “Font System” to manage mapping “letter codes → objects representing them in 3D views”. This implies that not only does the font system have its own *built-in* font, but it can use external fonts too, including *PostScript Type 1*, *OpenType* and *TrueType* fonts. And last but not least, it can use any objects existing in the current .blend file as letters.

Texts in Bender allow you to create/render 2D or 3D text, shaded as you want, with various advanced layout options (like justifying and frames), as we will see below. By default, letters are just flat filled surfaces, exactly like any closed 2D curve. But you can of course extrude them, and have them follow other curves.

Once you are happy with the shape of your text, you can convert it (with AltC, in Object mode), either to a curve, or directly to a mesh, allowing you to use all the powerful features of these types of objects on it.

(*Text Examples*) shows some examples of various fonts in action, including the “blue” font that has been applied to a curve path.

Notes

A maximum of **50000** characters is allowed per text object; however, be warned that the more characters a single text object has, the slower the object will respond interactively.

As you can see when you switch between Object and Edit modes, the Font panel remains the same. This means that its settings can be applied equally in both modes, and implies that you cannot apply them to just a part of the mesh. So font, size, and so on are common to all letters in a Text object. There is just one exception: the Bold/Italic buttons control properties specific to each letter (this is a way to use up to four different fonts in a text).

For optimum resource usage, only characters that are being used consume memory (rather than the entire character set).

Editing Text

Mode: Edit mode

Hotkey: see below

Text in Edit mode.

Editing text is quite different from other object types in Blender, and happens mainly in two areas. First, the 3D view, of course, where you type your text, and have a few shortcuts, e.g. for applying styles – note however that most Blender hotkeys you know in Edit mode do not exist for texts! The second place is the Button window (Editing context, F9), especially the Font panel.

The menu of the 3D view header has nearly no use, and there is no Specials menu. You have no transform or mirror tools, either, however you can apply the same modifiers to texts as for curves.

Editing Text is similar to using a standard text editor but is not as full-featured and has some differences:

Exit Edit mode
: ⇆ Tab doesn't insert a tab character in the text, but rather enters and exits Edit mode, as with other object types.

Copy
: To copy text to the buffer, use CtrlC or the Copy button in the tool shelf.

Cut and Copy
: To cut and copy text to the buffer, use CtrlX or the Cut button in the tool shelf.

Paste
: To paste text from the buffer, use CtrlV or the Paste button in the tool shelf.

Delete all text
: To completely erase or delete all text, use Ctrl← Backspace.

Home/End
: ↖ Home and ⇥ End move the cursor to the beginning and end of a line respectively.

Next/Previous word
: To move the cursor on a word's boundary, use Ctrl← or Ctrl→.

The text buffer does not communicate with the desktop. It only works within Blender. To insert text from outside Blender, see Inserting text below.

Inserting Text

You can insert text in three different ways: from the internal text buffer (Editing Text), or from a text file.

To load text from a text file, use the Text » Paste File tool. This will bring up a File Browser window for navigating to a valid UTF-8 file. As usual, be careful that the file doesn't have too many characters, as interactive response will slow down.

Special Characters

Mode: Edit mode

Menu: Text » Special Characters

There are a few special characters that are available using the Alt key or the Text menu in the 3D window header.

Here is a summary of these characters:

AltC:	Copyright (©)	AltR:	Registered trademark (®)
AltG:	Degrees (°)	AltX:	Multiply symbol (×)
AltS:	German "ss" (ß)	AltF:	Currency sign (¤)
AltL:	British Pound (£)	AltY:	Japanese Yen (¥)
Alt1:	Superscript 1 (¹)	Alt2:	Superscript 2 (²)
Alt3:	Superscript 3 (³)	Alt.:	Circle
Alt?:	Spanish question mark (¿)	Alt!:	Spanish exclamation mark (¡)
Alt<:	Left double quotation mark («)	Alt>:	Right double quotation mark (»)

All the characters on your keyboard should work, including stressed vowels and so on. If you need special characters (such as accented chars, which are not there on a US keyboard) you can produce many of them using a combination of two other characters. To do so, type the main char, press Alt← Backspace, and then press the desired "modifier" to produce the special character. Some examples are given below:

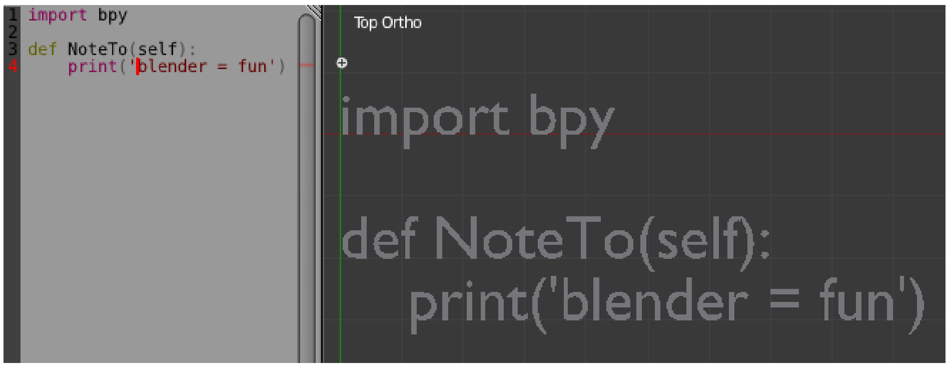
A, Alt← Backspace, ~:	ã	A, Alt← Backspace, ':	á	A, Alt← Backspace, `:	à
A, Alt← Backspace, O:	å	E, Alt← Backspace, ":	ë	O, Alt← Backspace, /:	ø

Convert text to text object

[[File:ConvertTextToTextObject.png|right|250px]Converting text to a text object.] An easy way to get text into Blender is to type it in The Text Editor. It is suggested to do this with a split window as you will be able to see the 3D view port and text editor at the same time. In the Text Editor select *Text > Create Text Block*. Then begin typing. When finished, select *Edit >> Text to 3D Object >> One Object* or *One Object per Line* depending on your needs. It is also possible to load a text file via *Text >> Open Text Block* which can be useful for importing large amounts of text at once.

3D Mesh

It is possible to convert a Text Object to a 3D Mesh object. This can be useful so that you may edit the vertices in Edit Mode, but you will lose the ability to edit the text itself. To do this, go to Object Mode and select your Text Object. Press AltC and select *Mesh From Curve/Meta/Surf/Text*. Now you can return to Edit Mode and manually edit the vertices. They are usually a bit messy, so it may be useful to use a *Limited Dissolve* deletion or *Remesh* Object Modifier at a low threshold to clean up your mesh.



```
import bpy

def NoteTo(self):
    print('blender = fun')
```

```
import bpy

def NoteTo(self):
    print('blender = fun')
```

Left: normal text. Right: the converted text object.

Text Selection

Text in Edit mode.

In Edit mode, your text has a white cursor, and as in any text editor, it determines where new chars will be inserted. You move this cursor with the arrow keys (→/↓/←/↑) or Page up/Page down and ↖ Home/⇥ End keys.

Hold ⇧ Shift while using the arrow keys to select a part of the text. You can use it to specify different materials, the normal/bold/italic state, and little else.

Formatting Text

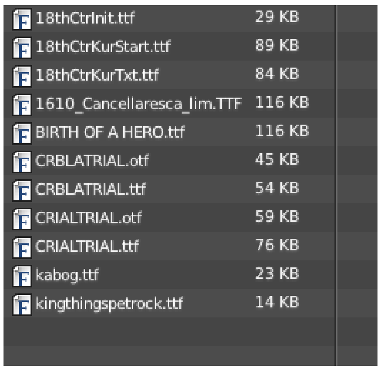
Fonts

Mode: Edit mode

Panel: Font (Editing context, F9)

The Font panel has several options for changing the look of characters.

Loading and Changing Fonts



18thCtrInit.ttf	29 KB
18thCtrKurStart.ttf	89 KB
18thCtrKurTxt.ttf	84 KB
1610_Cancellaresca_lim.TTF	116 KB
BIRTH OF A HERO.ttf	116 KB
CRBLATRIAL.otf	45 KB
CRBLATRIAL.ttf	54 KB
CRIALTRIAL.otf	59 KB
CRIALTRIAL.ttf	76 KB
kabog.ttf	23 KB
kingthingspetrock.ttf	14 KB

Loading a Type 1 font file.

Blender comes with a *built-in* font by default and is displayed in each of the four font style choosers. The *built-in* font is always present and shows in this list as “`Bfont`”. The first icon contains a drop-down list displaying currently loaded fonts. Select one for each font style.

To load a different Font, click one of the Load buttons in the Font panel and navigate to a *valid* font. The File Browser window will give all valid fonts a capital F icon, as seen in *Loading a Type 1 font file.*

Unix note
Fonts are typically located under `/usr/lib/fonts`, or some variant like `/usr/lib/X11/fonts`, but not always. They may be in other locations as well, such as `/usr/share/local` or `/usr/local/share`, and possibly related sub-trees.

If you select a font that Blender can't understand, you will get the error "`Not a valid font`".

Remember the same font will be applied to all chars with same style in a text, but that a separate font is required for each style. For example, you will need to load an *Italics* font in order to make characters or words italic. Once the font is loaded you can apply that font "Style" to the selected characters or the whole object. In all, you would need to load a minimum of four different types of fonts to represent each style (**Normal**, **Italic**, **Bold**, **Bold-Italic**).

It is important to understand that Blender does not care what font you load for "normal", "bold", etc., styles. This is how you can have up to four different fonts in use in the same text – but you have to choose between different styles of the same font, or different fonts. Blender has a number of typographic controls for changing the style and layout of text, found in the Font panel.

Size and Shear

Size
: Controls the size of the whole text (there is no way to control each char size independently). Note however that chars with different fonts (different styles, see below) might have different visible sizes.

Shear: 'blender' has a shear value of 1,
'2.59' a shear value of 0.

Shear
: Controls the inclination of the whole text. Even if this seems similar to italics style, *this is not the same thing*!

Objects as Fonts

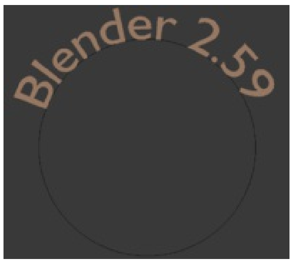
You can also "create" your own "font" inside Blender! This is quite a complex process, so let's detail it:

- First, you must create your chars. Each char is an object *of any type* (mesh, curve, meta…). They all must have a name following the schema: `common prefix` followed by the `char name` (e.g. "`font.a`", "`font.b`", etc.).
- Then, for the Text object, you must enable the Dupli Verts button (Object context – F7 –, Anim Settings panel).
- Back in Editing context (F9), in the Font panel, fill the Ob Family field with the *common prefix* of your "font" objects.

Now, each time a char in your text matches the *suffix part* of a "font" object's name, this object is duplicated on this char. *The original chars remain visible*. The objects are duplicated so that their center is positioned at the *lower right corner* of the corresponding chars.

Text on Curve

With the curve modifier you can let text follow a curve.


Text on a curve.

In (*Text on curve*) you can see a text deformed by a curve (a 2D Bézier circle).

To apply the curve modifier, the text object first has to be converted to a mesh, using AltC and click mesh.

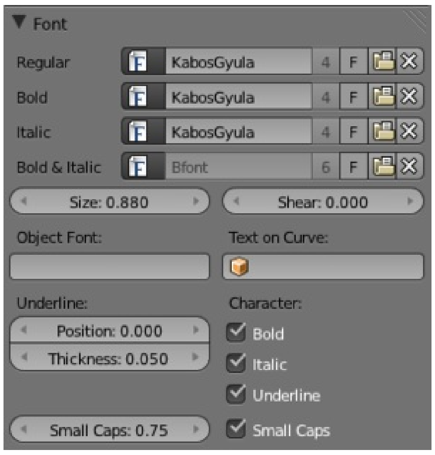
Note
There is also a Text on Curve feature, but the curve modifier offers more options.

Underline

Underline
: Toggled with the Underline button before typing. Text can also be set to Underlined by selecting it then using the Bold button in the Tool Shelf.

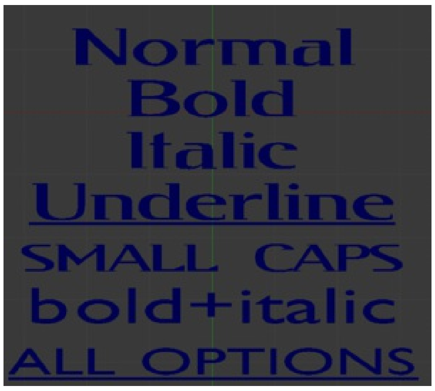
  Position
  : This allows you to shift vertically the position of the underline.

  Thickness
  : This controls the thickness of the underline.



Check a character option to type *e.g.* bold text.

Character



Bold text.

Bold
: Toggled with the Bold button before typing. Text can also be set to Bold by selecting it then using the Bold button in the Tool Shelf.

Italics
: Toggled with the Italic button before typing. Text can also be set to Italic by selecting it then using the Italic button in the Tool Shelf.

Underline
: Enables underlining, as controlled by the Underline settings above.

Small Caps
: Text is entered as small capital letters.

Blender's Bold and Italic buttons don't work the same way as other applications, as they also serve as placeholders for you to load up other fonts manually, which get applied when you define the corresponding style; see above.

To apply the Bold/Italics/Underline attribute to a set of characters, you either turn on Bold/Italics/Underline prior to typing characters, or highlight (select) first and then toggle Bold/Italics/Underline.

Setting Case

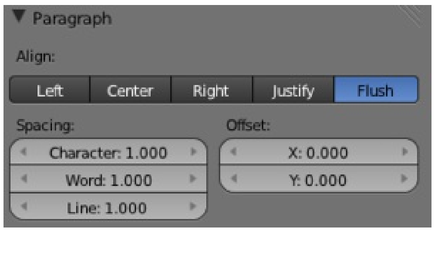
You can change the text case by selecting it then clicking the To Upper or To Lower in the tool shelf.

Enable the Small Caps option to type characters as small caps.

The size of the Small Caps can be changed with the Small Caps Scale setting. Note that the Small Caps Scale is applied to all Small Caps formatted characters the same way.

Paragraph

The Paragraph Panel has settings for the alignment and spacing of text.



The paragraph tab.

Align

Left
: Aligns text to left of frames when using them, else uses the center point of the Text object as the starting point of the text (which extends to the right).

Center
: Centers text in the frames when using them, else uses the center point of the Text object as the mid-point of the text (which extends equally to the left and right).

Right
: Aligns text to right of frames when using them, else uses the center point of the Text object as the ending point of the text (which extends to the left).

Justify
: Only flushes a line when it is **terminated** by a wordwrap (**not** by ↵ Enter), it uses *whitespace* instead of *character spacing* (kerning) to fill lines.

Flush
: **Always** flushes the line, even when it's still being entered; it uses character spacing (kerning) to fill lines.

Both Justify and Flush only work within frames.

Spacing

Character
: A factor by which space between each character is scaled in width

Word
: A factor by which whitespace between words is scaled in width. You can also control it by pressing Alt← or Alt→ to decrease/increase spacing by steps of **0.1**.

Line
: A factor by which the vertical space between lines is scaled.

Offset

X offset and Y offset
: These settings control the X and Y offset of the text, regarding its "normal" positioning. Note that with frames, it applies to all frames' content.

Shape

Mode: Object or Edit modes

Panel: Curve and Surface (Editing context, F9)

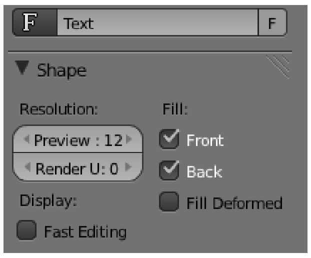
As you can see in the Curve and Surface panel, texts have most of the same options as curves.

Resolution

Preview
: The resolution in the viewport.

Render
: The resolution on the render.

Fast Editing
: Disables curve filling while in edit mode.


Text
Shape
Resolution:
Preview : 12
Render U: 0
Display:
[ ] Fast Editing
Fill:
[✓] Front
[✓] Back
[ ] Fill Deformed

The shape settings.

Fill

The fill options control how the text curves are filled in when text is Extruded or Beveled in the Geometry Panel.

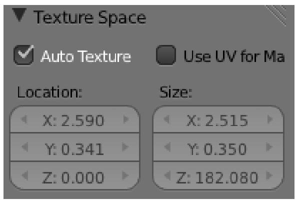
Front
: Fills in the front side of the surface.

Back

Fills in the back side of the surface.

Fill Deformed

Fills the curves after applying shape keys and modifiers.


Texture Space
Auto Texture ✓ Use UV for Ma
Location: X: 2.590 Y: 0.341 Z: 0.000
Size: X: 2.515 Y: 0.350 Z: 182.080

Texture Settings.

Textures

Use UV for Mapping

Use UV values as generated texture coordinates.

Auto Texture Space

Adjusts the active object's texture space automatically when transforming object.

Geometry

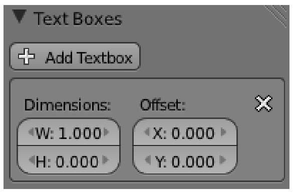
Text objects have all the extrusion features of curves.

Text Editing

Text Boxes

Mode: Object or Edit modes

Panel: Font (Editing context, F9)



Text frame.

Text “Boxes” allow you to distribute the text amongst rectangular areas within a single text object. An arbitrary number of freely positionable and re-sizable text frames are allowed per text object.

Text flows continuously from the lowest-numbered frame to the highest-numbered frame with text inside each frame word-wrapped. Text flows between frames when a lower-numbered frame can’t fit any more text. If the last frame is reached, text overflows out of it.

Text frames are very similar to the concept of *frames* from a desktop publishing application, like Scribus. You use frames to control the placement and flow of text.

Frames are controlled in the Text Boxes panel.

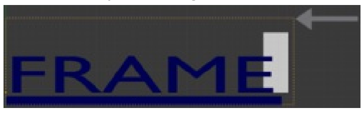
Frame size

By default the first frame for a new text object, and any additional frames, has a size of **zero** for both Width and Height, which means the frame is initially not visible.

Frames with a width of **0.0** are ignored completely during text flow (no wordwrap happens), and frames with a height of **0.0** flow forever (no flowing to the next text frame).

In order for the frame to become visible, the frame’s Width must be greater than **0.0**.

Note
Technically the height is never actually **0.0** because the font itself always contributes height.



Frame width.

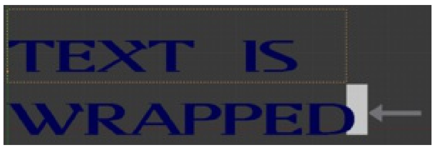
(*Frame width*) is a text object with a width of **5.0**. And because the frame width is greater than **0.0** it is now visible and is drawn in the active theme color as a dashed rectangle. The text has overflowed because the text has reached the end of the last frame, the default frame.

Adding/Deleting a Frame

To add a frame click the Add Textbox button on the Text Boxes panel. A new frame is inserted just after (in text flow order) the current one, with its attributes (position and size). Be sure to modify the offset for the new frame in the X and/or Y fields. Just an X modification will create a new column.

To delete the current frame, click the Delete button. Any text in higher frames will be re-flowed downward into lower frames.

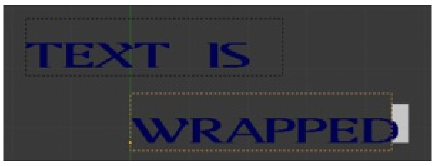
Example: Text Flow



wrapping

With two or more frames you can organize text to a finer degree. For example, create a text object and enter “`Blender is super duper`”. This text object has a frame; it just isn’t visible because its Width is **0.0**.

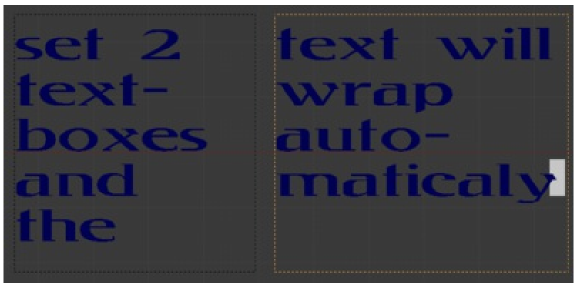
Set the width to **5.0**. The frame is now visible and text is wrapping according to the new width, as shown in (*Text 2*). Notice that the text has overflowed out of the frame. This is because the text has reached the end of the last frame, which just happens to be the default/initial frame.



text flowing from box 1 to box 2

When we add another frame and set its width and height, the text will flow into the new frame.

Example: Multiple columns



Text 5.

To create two columns of text just create a text object and adjust the initial frame’s Width and Height to your requirements, then insert a new frame. The new frame will have the same size as the initial frame. Set the X position to something greater or less than the width of the initial frame; see (*Text 5*).

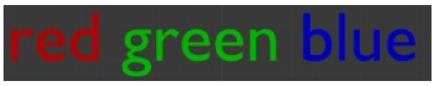
Assigning Materials

Mode: Edit mode

Panel: Link and Materials (Editing context, F9)

Each character can have a different Material index in order to have different materials on different characters.

You can assign indices either as you type, or after by selecting blocks of text and clicking on the Assign button in the Materials panel.



Red Green Blue.

For example, to create (*Red Green Blue*) you would need to create three separate materials and three separate material indices. Each word would be assigned a Material index by selecting the characters for each word and clicking the Assign button. (*Red Green Blue*) is still one single Text object.

Meta Objects

Mode: Object or Edit modes

Hotkey: ⇧ ShiftA

Menu: Add » Meta

Meta objects are *implicit surfaces*, meaning that they are *not explicitly* defined by vertices (as meshes are) or control points (as surfaces are): they exist *procedurally*. Meta objects are literally mathematical formulas that are calculated on-the-fly by Blender.

A very distinct visual characteristic of metas is that they are fluid *mercurial*, or *clay-like* forms that have a “rounded” shape. Furthermore, when two meta objects get close to one another, they begin to interact with one another. They “blend” or “merge”, as water droplets do, especially in zero-g (which, by the way, makes them very handy for modeling streams of water when you don’t want to do a fluid simulation). If they subsequently move away from one another, they restore their original shape.

Each of these is defined by its own underlying mathematical structure, and you can at any time switch between them using the Active Element panel.

Typically Meta objects are used for special effects or as a basis for modeling. For example, you could use a collection of metas to form the initial shape of your model and then convert it to another object type (well, only meshes are available…) for further modeling. Meta objects are also very efficient for ray-tracing.

Note that Meta objects have a slightly different behavior in Object mode, as detailed below.

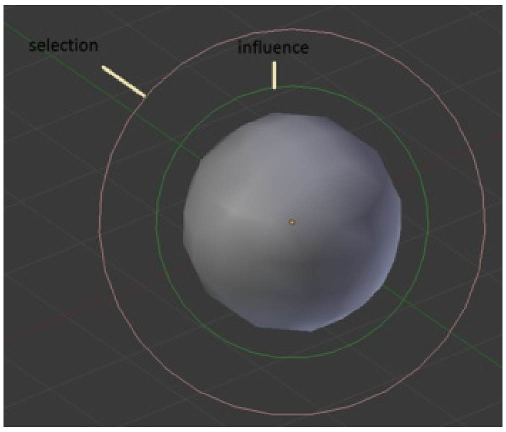
Primitives

There are five predefined meta “primitives” (or configurations) available in the Add » Meta sub-menu:

- Meta Ball adds a meta with a point underlying structure.
- Meta Tube adds a meta with a line segment underlying structure.
- Meta Plane adds a meta with a planar underlying structure.
- Meta Ellipsoid adds a meta with an ellipsoidal underlying structure.
- Meta Cube adds a meta with a volumetric cubic underlying structure.

Visualization

In Object mode, the calculated mesh is shown, along with a black “selection ring” (becoming pink when selected). To learn more about metas in Object mode, see below.

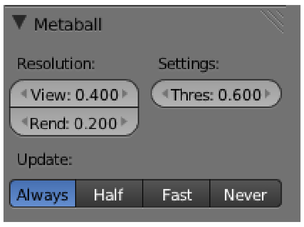


Meta Ball example.

In Edit mode (*Meta Ball example*), a meta is drawn as a mesh (either shaded or as black wireframe, but without any vertex of course), with two colored circles: a red one for selection (pink when selected), and a green one for a direct control of the meta’s stiffness (see below – light green when active). Note that except for the Scale (S) transformation, having the green circle highlighted is equivalent to having the red one.

Meta Ball Options

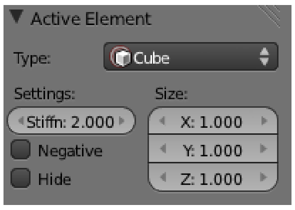
All Meta objects in a scene interact with each other. The settings in the MetaBall section apply to all meta objects. In Edit mode, the Active Element panel appears for editing individual meta elements.


Metaball

Resolution: View: 0.400, Rend: 0.200

Settings: Thres: 0.600

Update: [✓ Always] Half Fast Never


Active Element

Type: Cube

Settings: Stiffn: 2.000, Negative, Hide

Size: X: 1.000, Y: 1.000, Z: 1.000

global meta properties. individual meta properties.

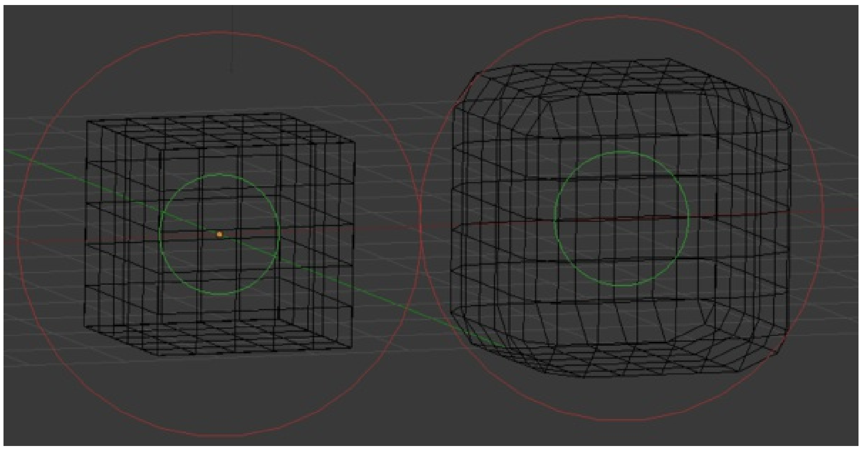
Resolution

The Resolution controls the resolution of the resultant mesh as generated by the Meta object.

View
: The 3D View resolution of the generated mesh. The range is from **0.05** (finest) to **1.0** (coarsest).

Render
: The rendered resolution of the generated mesh. The range is from **0.05** (finest) to **1.0** (coarsest).

One way to see the underlying mathematical structure is to lower the Resolution, increase the Threshold and set the Stiffness (see below) a fraction above the Threshold. (*Underlying structure*) is a (*Meta cube*) with the above mentioned configuration applied as follows: Resolution of **0.410**, Threshold of **5.0** and Stiffness a fraction above at **5.01**.


Left: Underlying structure, Right: the shape.

You can clearly see the underlying cubic structure that gives the meta cube its shape.

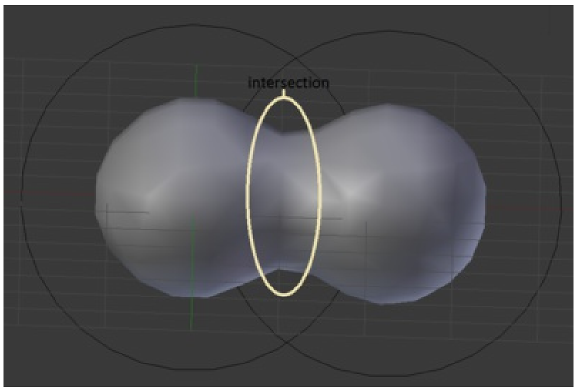
Threshold (Influence)

Mode: Object or Edit modes

Panel: MetaBall (Editing context, F9)

Threshold defines how much a meta's surface "influences" other metas. It controls the *field level* at which the surface is computed. The setting is global to a group of Meta objects. As the threshold increases, the influence that each meta has on each other increases.

There are two types of influence: **positive** or **negative**. The type can be toggled on the Active Element panel while in Edit mode, using the Negative button. You could think of **positive** as attraction and **negative** as repulsion of meshes. A negative meta will push away or repel the meshes of positive Meta objects.



Positive.

A *positive* influence is defined as an attraction, meaning the meshes will stretch towards each other as the *rings of influence* intersect. (*Positive*) shows two meta balls' *rings of influence* intersecting with a *positive* influence.

Notice how the meshes have pulled towards one another. The area circled in white shows the green *influence* rings intersecting.

Update

While transforming metas (grab/move, scale, etc.), you have four "modes" of visualization, located in the Update buttons group of the

MetaBall panel:

- Always – fully draw the meta during transformations.
- Half Res – During transformations, draw the meta at half its Wiresize resolution.
- Fast – Do not show meta mesh during transformations.
- Never – Never show meta mesh (not a very recommended option, as the meta is only visible at render time!).

This should help you if you experience difficulties (metas are quite compute-intensive…), but with modern computers, this shouldn't happen, unless you use many metas, or very high resolutions…

Meta Structure

Technical Details

A more formal definition of a meta object can be given as a *directing structure* which can be seen as the source of a static field. The field can be either positive or negative and hence the field generated by neighboring directing structures can attract or repel.

The implicit surface is defined as the surface where the 3D field generated by all the directing structures assume a given value. For example a meta ball, whose directing structure is a point, generates an isotropic (i.e. identical in all directions) field around it and the surfaces at constant field value are spheres centered at the directing point.

Meta objects are nothing more than mathematical formulae that perform logical operations on one another (AND, OR), and that can be added and subtracted from each other. This method is also called **Constructive Solid Geometry** (CSG). Because of its mathematical nature, CSG uses little memory, but requires lots of processing power to compute.

Underlying Structure

Mode: Edit mode

Panel: MetaBall tools (Editing context, F9), Transform Properties

Blender has five types of metas, each determined by its underlying (or directing) structure. In Edit mode, you can change this structure, either using the relevant buttons in the MetaBall tools panel, or the drop-down list in the Transform Properties panel (N). Depending on the structure, you might have additional parameters, located in both Transform Properties and MetaBall tools panels.

Ball (point, zero-dimensional structure)
: This is the simplest meta, without any additional setting. As it is just a point, it generates an isotropic field, yielding a spherical surface (this is why it is called Meta Ball or Ball in Blender).

Tube (straight line, uni-dimensional structure)
: This is a meta which surface is generated by the field produced by a straight line of a given length. This gives a cylindrical surface, with rounded closed ends. It has one additional parameter:

  - dx: The length of the line (and hence of the tube – defaults to **1.0**).

Plane (rectangular plane, bi-dimensional structure)
: This is a meta which surface is generated by the field produced by a rectangular plane. This gives a parallelepipedal surface, with a fixed thickness, and rounded borders. It has two additional parameters:

  - dx: The length of the rectangle (defaults to **1.0**).
  - dy: The width of the rectangle (defaults to **1.0**).

  Note that by default, the plane is a square.

Elipsoid (ellipsoidal volume, tri-dimensional structure)
: This is a meta which surface is generated by the field produced by an ellipsoidal volume. This gives an ellipsoidal surface. It has three additional parameters:

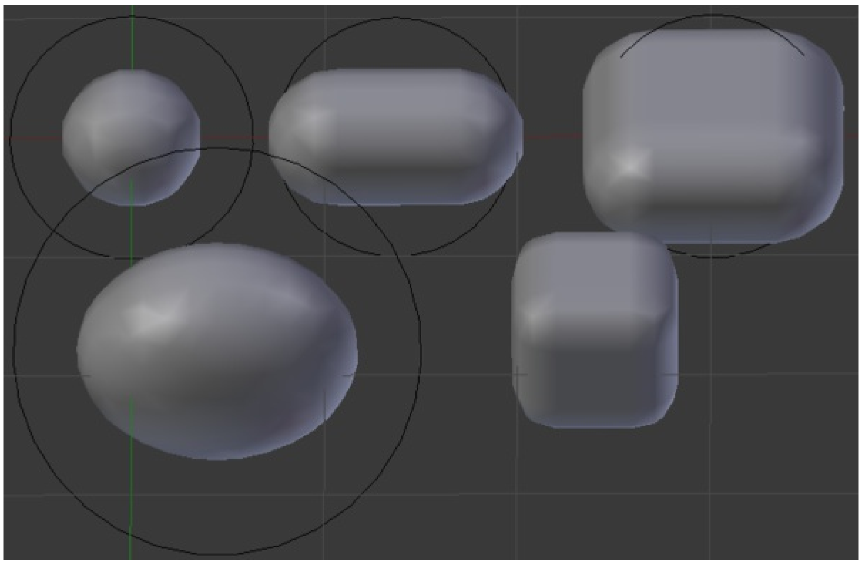
  - dx: The length of the ellipsoid (defaults to **1.0**).
  - dy: The width of the ellipsoid (defaults to **1.0**).
  - dz: The height of the ellipsoid (defaults to **1.0**).

  Note that by default, the volume is a sphere, producing a spherical meta, as the Ball option…

Cube (parallelepipedal volume, tri-dimensional structure)
: This is a meta which surface is generated by the field produced by a parallelepipedal volume. This gives a parallelepipedal surface, with rounded edges. As you might have guessed, it has three additional parameters:

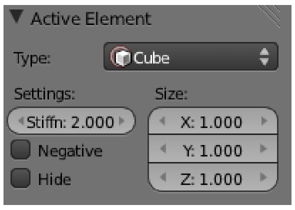
  - dx: The length of the parallelepiped (defaults to **1.0**).
  - dy: The width of the parallelepiped (defaults to **1.0**).
  - dz: The height of the parallelepiped (defaults to **1.0**).

  Note that by default, the volume is a cube.


the 5 meta primitives.

Editing Metas

When in Edit mode, the Active Element panel appears. These settings apply only to the selected meta element.


Active Element
Type: Cube
Settings: Stiffn: 2.000
Negative
Hide
Size: X: 1.000 Y: 1.000 Z: 1.000

the active element panel.

Meta Shape

The Type menu lets you change the shape of the meta object, as explained above.

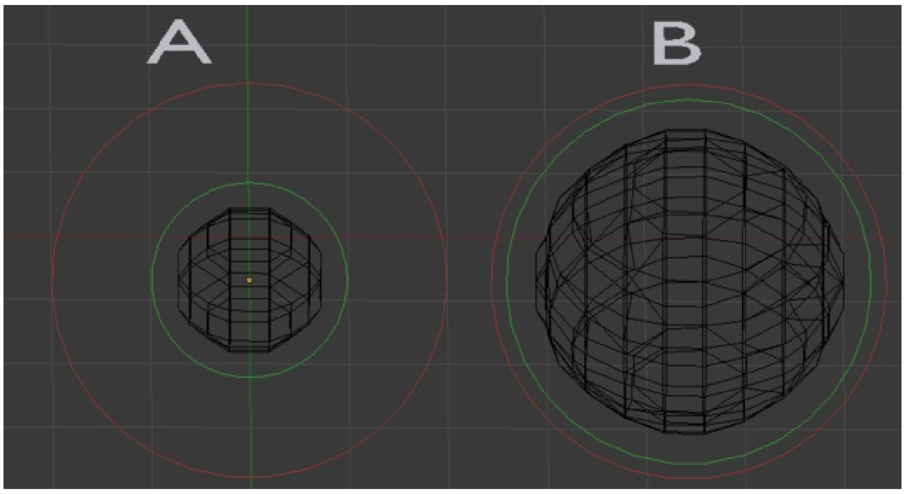
Stiffness

Together with Threshold, Stiffness controls the influencing range. While the threshold is common to all metas in the same object (or even the same object family), the stiffness is specific to each meta.

Scaling the inner green circle changes the Stiffness value. Stiffness defines how much the meta object is filled. This essentially defines how sensitive a meta is to being affected by other metas. With a low stiffness, the meta will begin to deform from further away. A higher value means the meta needs to be close to another one to begin merging.

When a Meta object comes within "range" of another meta, the two will begin to interact with each other. They don't necessarily need to intersect, and depending on the Threshold and Stiffness settings, they most likely won't need to. Stiffness is materialized by the *green ring*.

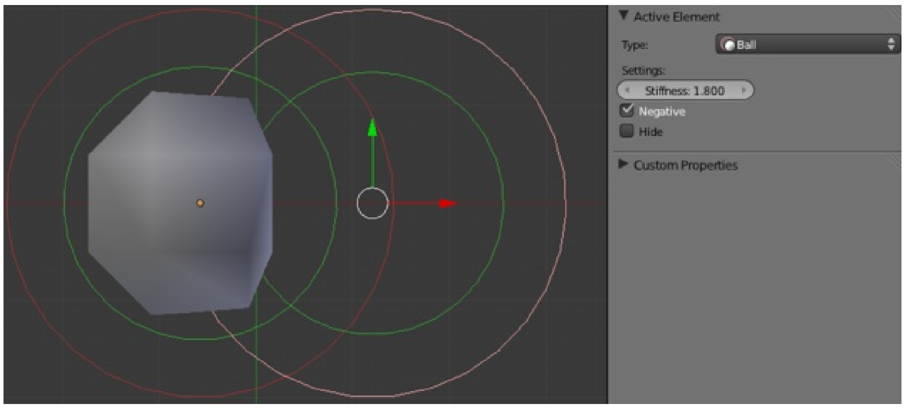
The range is from **0.0** to **10.0**. But to be visible, the Stiffness must be slightly larger than the Threshold value. You can also visually adjust the Stiffness ring by using the RMB to select it and activate Scale mode with S.



Stiffness.

In (*Stiffness*), the meta ball labeled "A", has a smaller Stiffness value than the one labeled "B". As you can see, the *green ring* radius is different for each of them.

Negative Influence


Active Element
Type: Ball
Settings: Stiffness: 1.800
✓ Negative
Hide
Custom Properties

Negative.

The opposite effect of a *positive* influence would be a *negative* influence: the objects repel each other. (*Negative*) shows a meta ball and a meta plane where the first is negative and the second, positive. Notice how the negative meta is not visible: only the surrounding circles appear. This is how Blender indicates that the object is negative.

Moving the sphere to the plane causes the plane’s mesh to “cave in” or collapse inward. If you move the plane away from the sphere, the plane’s mesh will restore itself.

To make a meta *negative*, just select the meta in edit mode, and check *negative* in the *active element* panel.

Hiding Elements

As in Object mode, you can hide the selected meta(s), and then reveal what was hidden. This is very handy for cleaning your views up a bit… Note that the two red and green rings always remain visible in Edit mode, as well as the select circle (in Object mode…).

To hide the current selection, use H, the Hide toggle button in the MetaBall tools, or the Metaball » Hide MetaElems » Hide Selected menu option.

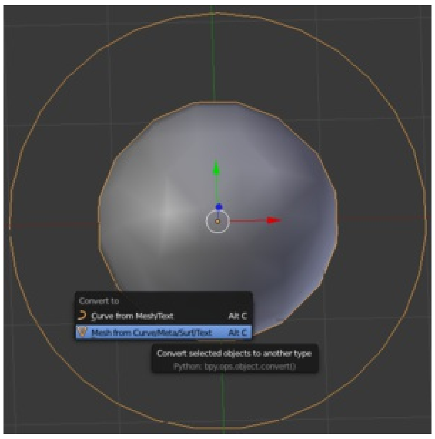
To hide everything but the current selection, hit ⇧ ShiftH or use Metaball » Hide MetaElems » Hide Deselected.

To reveal what was hidden, use AltH, or the relevant option in the same Metaball » Hide MetaElems menu. You can also un-toggle the Hide button in the (MetaBall tools panel).

Deleting Elements

There is no Erase menu for metas, just a confirmation pop-up asking you if you want to delete the selected metas. Clear and simple!

Conversion


Convert to
Curve from Mesh/Text Alt C
Mesh from Curve/Meta/Surf/Text Alt C
Convert selected objects to another type
Python: bpy.ops.object.convert()
the convert menu

You can only convert metas to meshes, but here you have the option to keep the original Meta object (i.e. create a new Mesh one, instead of a “real” conversion…). Note that the resolution used for the new mesh is the Wiresize one, not the Rendersize one.

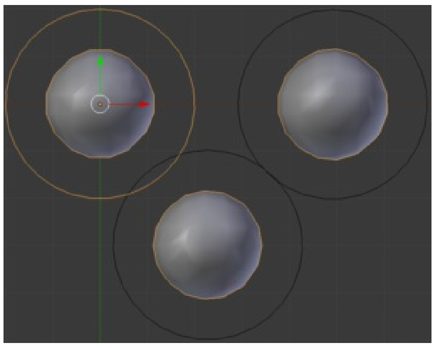
To convert the meta, press AltC in Object mode, and select *mesh*

Object Families

Meta objects have different behavior in Object mode than other object types – they can be “regrouped” into so-called “families”.

A “family” is a way to regroup several meta objects, producing something very similar to having several metas inside the same object.

A family is defined by the left part of an object’s name (the one before the dot). Remember, an object’s name is the one in the “OB” field, in most panels, **not** the “MB” field, which is the meta datablock’s name… For example, the *family* part of “`MetaPlane.001`” is “`MetaPlane`”. Each meta object in the same “family” is associated with one another as discussed below.

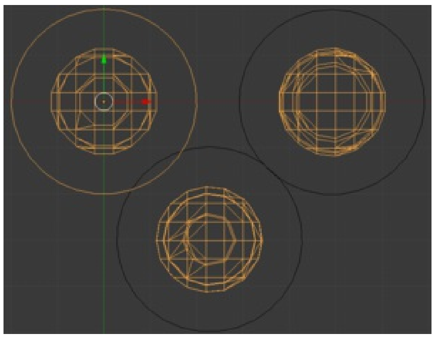

Meta ball base.

Families of metas are controlled by a *base* Meta object which is identified by an Object name **without** a right part. For example, if we have five metas called “`MetaThing`”, “`MetaThing.001`”, “`MetaThing.002`”, “`MetaThing.003`” and “`MetaThing.004`”, the *base* Meta object would be “`MetaThing`”.

The *base* Meta object determines the basis, the resolution, the threshold, *and* the transformations. It also has the material and texture area. The *base* meta is effectively the parent of (or perhaps a better word to use is “the owner of”) the other metas in the group (i.e. it is as if the other metas were “included” or joined into the base one).

Examples

(*Meta ball base*) shows the *base* meta labeled “`B`”. The other two Meta objects are *children*. Children’s selection rings are always black, while the group’s mesh is orange. Because the metas are grouped, they form a unified mesh which can always be selected by selecting the mesh of any meta in the group. For example, in the example (*Meta ball base*), only the lower sphere (the parent) has been selected, and you see that both the parent’s mesh *and* all of the children’s meshes are now highlighted.


Scaling the “base”.

The *base* Meta object controls the **polygonalization** (mesh structure) for the group, and as such, also controls the polygonalization for the children (*non-base*) metas. If we transform the *base* meta, the children’s polygonalization changes. However, if we transform the children, the polygonalization remains unchanged.

Hints

This discussion of “polygonization” *doesn’t* mean that the various meshes don’t deform towards or away from each other (meta objects always influence one another in the usual way, whether or not they are members of the same family). Rather, it means that the underlying mesh structure changes only when the *base* object transforms. For example, if you scale the *base*, the children’s mesh structure changes. In (*Scaling the “base”*), the *base* has been scaled down, which has the effect of scaling the mesh structure of each of the children. As you can see, the children’s mesh resolution has increased, while the *base* decreased. *The children did not change size!*

Empties

The "Empty" is a null object. It contains no real Geometry, but can be used as a handle for many purposes.

Settings

Plain Axes
: Draws as six lines, initially with one pointing in each of the +X,-X,+Y,-Y,+Z,and -Z axis directions.

Arrows
: Draws as arrows, initially pointing in the positive X,Y, and Z axis directions, each with a label.

Single Arrow
: Draws as a single arrow, initially pointing in the +Z axis direction.

Circle
: Draws as a circle initially in the XZ plane.

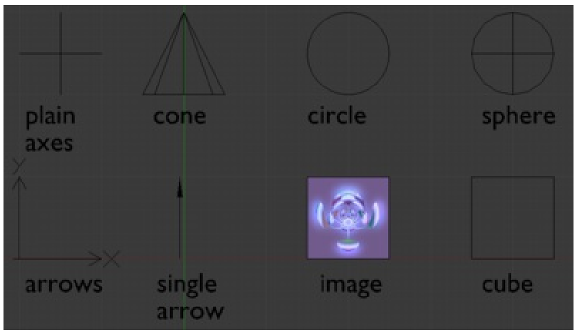
Cube
: Draws as a cube, initially aligned to the XYZ axes.

Sphere
: Draws as an implied sphere defined by 3 circles. Initially, the circles are aligned, one each, to the X, Y, and Z axes.

Cone
: Draws as a cone, initially pointing in the +Y axis direction.

Image
: Empties can display images. This can be used to create reference images, including blueprints or character sheets to model from, instead of using background images. The image is displayed regardless of the 3D display mode. The settings are the same as in Background Image Settings
Note: While alpha-images can be used, there is a known limitation with object draworder, where alphas won't always drawon top of other objects when unselected.

Size
: Controls the local size of the empty. This does not change its scale, but simply resizes the shape.



The eight different empty draw types as seen from the top view

Usage and functions

Empties can serve as transform handles which cannot be edited and do not render. Empties are important and useful objects. Some examples of ways to use them include:

Parent object for a group of objects

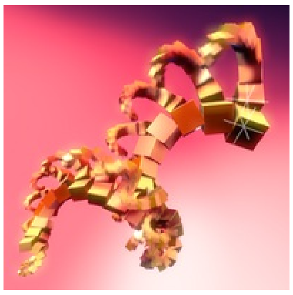
- An Empty can be parented to any number of other objects - This gives the user the ability to control a group of objects easily, and without affecting a render.

Target for constraints

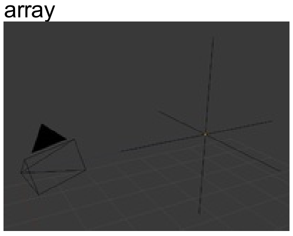
- An empty can also be used as a target for normal, or bone constraints.
- This gives the user far more control; for instance, a rig can easily be set up to enable a camera to point towards an empty using the ***Track to*** constraint

Array offset

- An empty can be used to offset an array modifier, meaning complex deformations can be achieved by only moving a single object.


An example of an empty

being used to control an array


An example of an empty being used to control the track to constraint

Other common uses.

- Placeholders
- Rigging controls
- DOF distances
- Reference Images

Group Instances

Groups are covered fully here: Grouping and Parenting.

If a group already exists in the scene, an instance of one of those groups can be created from the Add menu under add » group instance » group.

These group proxies are controlled/owned by an additional empty object, and hence are not editable (i.e. have no Edit mode) – in fact, they behave a bit as if all object copies of the group were children of the empty). So you have all the editing options of objects (you can move/scale/rotate them, etc.).

Groups and Metas
There seems to be a bug with Meta objects in group proxies: their meshes are invisible; only the circles are drawn…

Visualization

Group “proxies” are drawn in black (unless Solid or Shaded draw mode, of course). The selection of “real” members of the group is reflected in their “proxies”. When the proxy itself is selected, the empty turns pink, and the other parts, purple.

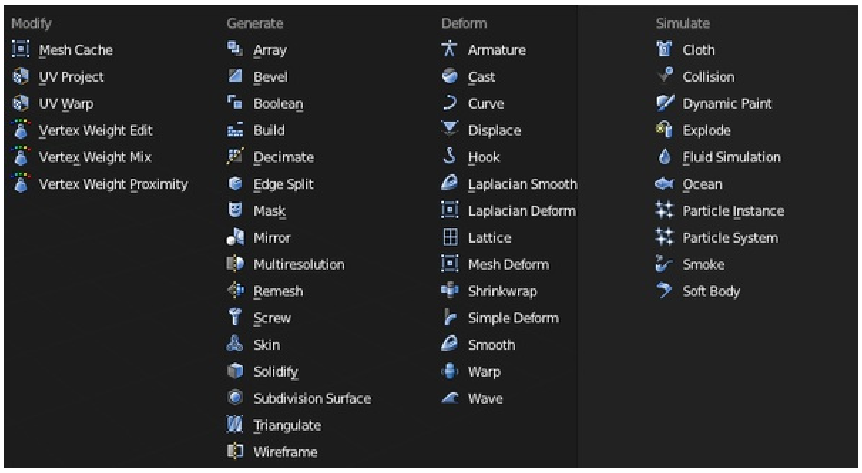
The only options of these “group proxies” are the same as the ones for empties visualization.

Modifiers

Mode: Any mode

Panel: Modifiers

Modifiers are automatic operations that affect an object in a non-destructive way. With modifiers, you can perform many effects automatically that would otherwise be tedious to do manually (such as subdivision surfaces) and without affecting the base topology of your object. Modifiers work by changing how an object is displayed and rendered, but not the actual object geometry. You can add several modifiers to a single object to form a Modifier Stack and you can Apply a modifier if you wish to make its changes permanent.



There are four types of modifiers:

Modify

The Modify group of modifiers are tools a bit similar to the Deform Modifiers (see below), but which do not directly affect the shape of the object; rather they affect some other data, like vertex groups…

Mesh Cache
    Apply animated mesh data (from external file) to a mesh.
UV Project
    Project UV coordinates on your mesh.
UV Warp
    Dynamically edit the UV coordinates on your mesh.
Vertex Weight
    Edit a vertex group of your mesh, in various ways.

Generate

The Generate group of modifiers are constructive tools that either change the general appearance of or automatically add new geometry to an object.

Array
    Create an array out of your basic mesh and similar (repeating) shapes.
Bevel
    Create a bevel on a selected mesh object.
Boolean
    Combine/subtract/intersect your mesh with another one.
Build
    Assemble your mesh step by step when animating.
Decimate
    Reduce the polygon count of your mesh.
Edge Split
    Add sharp edges to your mesh.
Mask
    Allows you to hide some parts of your mesh.
Mirror
    Mirror an object about one of its own axes, so that the resultant mesh is symmetrical.
Multiresolution
    Sculpt your mesh at several levels of resolution.
Remesh
    Can fix heavily triangulated meshes, and other issues, with careful Threshold adjustments.
Screw
    Generate geometry in a helix-pattern from a simple profile. Similar to the Screw tool in the mesh editing context.
Skin
    Automatically generate topology.
Solidify
    Give depth to mesh faces.
Subdivision Surface

Subdivides your mesh using Catmull-Clark or Simple algorithms.

Triangulate
: Converts all faces to Triangles.

Wireframe
: Converts all faces into a wireframe (included in trunk atm).

Deform

The Deform group of modifiers only change the shape of an object, and are available for meshes, and often texts, curves, surfaces and/or lattices.

Armature
: Use bones to deform and animate your object.

Cast
: Shift the shape of a mesh, surface or lattice to a sphere, cylinder or cuboid.

Curve
: Bend your object using a curve as guide.

Displace
: Deform your object using a texture.

Hook
: Add a hook to your vertice(s) (or control point(s)) to manipulate them from the outside.

Laplacian Smooth
: Allows you to reduce noise on a mesh's surface with minimal changes to its shape.

Laplacian Deform
: allows you to pose a mesh while preserving geometric details of the surface.

Lattice
: Use a Lattice object to deform your object.

Mesh Deform
: Allows you to deform your object by modifying the shape of another mesh, used as a “Mesh Deform Cage” (like when using a lattice).

Shrinkwrap
: Allows you to shrink/wrap your object to/around the surface of a target mesh object.

Simple Deform
: Applies some advanced deformations to your object.

Smooth
: Smooth the geometry of a mesh. Similar to the Smooth tool in the mesh editing context.

Warp
: Warp a mesh by specifying two points the mesh stretches between.

Wave
: Deform your object to form (animated) waves.

Simulate

The Simulate group of modifiers activate simulations. In most cases, these modifiers are automatically added to the modifiers stack whenever a Particle System or Physics simulation is enabled, and their only role is to define the place in the modifiers stack used as base data by the tool they represent. Generally, the attributes of these modifiers are accessible in separate panels.

Cloth
: Simulates the properties of a piece of cloth. It is inserted in the modifier stack when you designate a mesh as Cloth.

Collision
: Simulates a collision between objects.

Dynamic Paint
: Makes an object or a particle system paint a material onto another object.

Explode
: Blows up your mesh using a particle system.

Fluid
: The object is part of a fluid simulation… The modifier added when you designate a mesh as Fluid.

Particle Instance
: Makes an object act similar to a particle but using the mesh shape instead.

Particle System
: Represents a particle system in the stack, so it is inserted when you add a particle system to the object.

Smoke
: Simulates realistic smoke.

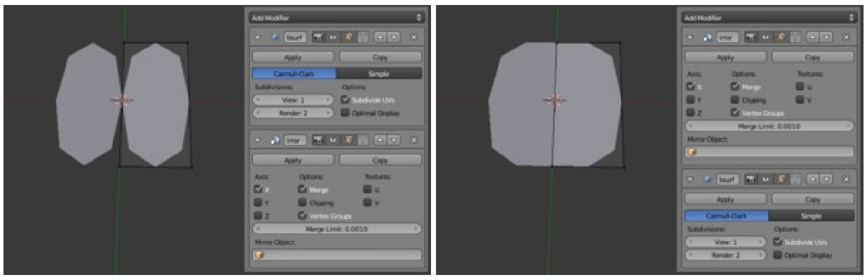
Soft Body
: The object is soft, elastic… Modifier added when you designate a mesh as Softbody.

Ocean
: Quickly creates a realistic, animated ocean.

Modifier

A modifier is defined as the application of a “process” or “algorithm” upon objects. They can be applied interactively and non-destructively in just about any order the users chooses. This kind of functionality is often referred to as a “modifier stack” and is found in several other 3D applications.

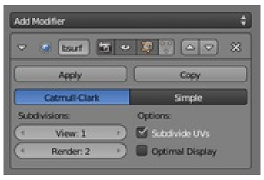
Modifiers are added from the Modifiers menu. Some tools and scripts, for example Decimate and Solidify, have been migrated from its previous location and changed into a modifier. In a modifier stack the order in which modifiers are applied has an effect on the result. Fortunately modifiers can be rearranged easily by clicking the convenient up and down arrow icons. For example, (*Stack ordering*) shows SubSurf and Mirror modifiers that have switched places.


Stack ordering

In the first example, the Mirror modifier is the last item in the stack. The result looks like two surfaces. In the other example the mirror modifier is the first item in the stack and the result is a single merged surface.

You can see that the results look different from the previous. This means that the stack order is very important in defining the end results.

Interface



Panel Layout (Subsurf as an example)

Each modifier has been brought in from a different part of Blender, so each has its own unique settings and special considerations. However, each modifier’s interface has the same basic components, see (*Panel Layout (Subsurf as an example)*).

At the top is the panel header. The icons each represent different settings for the modifier (left to right):

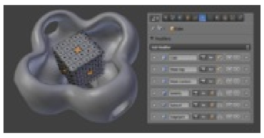
1. Arrow — Collapse modifier to show only the header.
2. Modifier icon and a box for the name of this modifier — default being the name of the modifier itself. It is unique amongst other modifiers of the same type.
3. Camera — Display modifier effect during render time.
4. Eye — Display modifier effect in the 3D view.
5. Box — Shows modifier effect in Edit mode. This button may not be available depending on the type of modifier.
6. Triangle — Applies modifier to editing cage in Edit mode. This icon materializes the Cage mode.
7. Up arrow — Moves modifier up in the stack.
8. Down arrow — Moves modifier down in the stack.
9. Cross — Removes the modifier from the stack.

Below the header are two buttons:

1. Apply – Makes the modifier real.
2. Copy – Creates a copy of the modifier at the base of the stack.

And below these buttons is a sub panel with settings for individual modifiers.

Stack


In this example a simple subdivided cube has been transformed into a rather complex object using a stack of modifiers. (.blend)

To add a modifier you add it to the *stack*. Once added (always at the bottom of the stack), they can be rearranged to your liking.

Some modifiers can only be applied to certain object types. This is indicated by the panel filtering the Add Modifier button on the Modifiers panel. Only modifiers that can be applied are shown in this listbox button. Mesh objects can have all available modifiers added, while, for example, Lattice objects type objects can only have a few.

Mesh Cache Modifier

Mode: Any mode

Panel: Modifiers

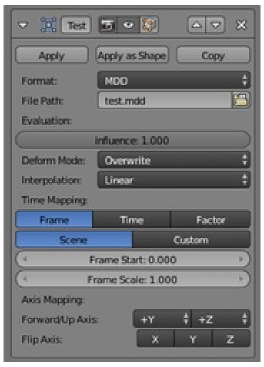
Description

The Mesh Cache modifier is used so animated mesh data can be applied to a mesh and played back, deforming the mesh.

This works in a similar way to shape-keys but is aimed at playing back external files and is often used for interchange between applications.

When using this modifier, the vertex locations are overwritten.

Options



Format: MDD
File Path: test.mdd
Influence: 1.000
Deform Mode: Overwrite
Interpolation: Linear
Frame Start: 0.000
Frame Scale: 1.000
Forward/Up Axis: +Y +Z
Flip Axis:

Mesh Cache modifier

Format
: The input file format (currently **MDD** and **PC2** are supported).

File Path
: Path to the cache file.

Influence
: Factor to adjust the influence of the modifiers deformation, useful for blending in/out from the cache data.

Deform Mode
: This setting defaults to 'Overwrite' which will replace the vertex locations with those in the cache file.
However you may want to use shape-keys, for example, and mix them with the mesh-cache. In this case you can select the 'Deform' option which integrates deformations with the mesh-cache result.
Note - this feature is limited to making smaller, isolated edits and won't work for re-posing limbs, for example.

Interpolation
: None or Linear which will blend between frames; use linear when the frames in the cache file don't match up exactly with the frames in the blend file.

Time Mapping:

Time Mode
: Select how time is calculated.

- **Frame:** Allows you to control the frames, which will ignore timing data in the file but is often useful since it gives simple control.
- **Time:** Evaluates time in seconds, taking into account timing information from the file (offset and frame-times).
- **Factor:** Evaluates the entire animation as a value from [0 - 1].

Play Mode
: Select how playback operates.

- **Scene:** Use the current frame from the scene to control playback.
  - **Frame Start:** Play the cache starting from this frame.
  - **Frame Scale:** Scale time by this factor (applied after the start value).
- **Custom:** Control animation timing manually.
  - **Evaluation Value:** Property used for animation time, this gives more control of timing - typically this value will be animated.

Axis Mapping:

Axis transformation for the input coordinates.

Forward/Up Axis
: The axis for forward and up used in the source file.
Often different applications have different axis defaults for up/down front/back, so it's common to have to switch these on import.

Flip Axis

- In rare cases you may also need to flip the coordinates on an axis.

Hints

- Both MDD and PC2 depend on the vertex order on the mesh remaining unchanged; this is a limitation with the method used so take care not to add/remove vertices once this modifier is used.

UV Project Modifier

Mode: Any mode

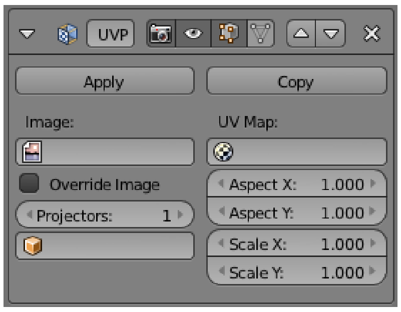
Panel: Modifiers (Generate)

Projecting the Blender logo onto Suzanne.

The UV Project Modifier acts like a slide projector. It emits a UV map from the -Z axis of 'Projector' objects, and applies it to the object as the "light" hits it. It can optionally override the object's face texture.

- File:Uvproject.blend

Options



UV layer
: Which UV layer to modify. Defaults to the active rendering layer.

Image
: The image associated with this modifier. Not required; you can just project a UV for use elsewhere. *Override Image*, below, defines how the image is used.

Override Image

- When true, the Face Texture of all vertices on the mesh is replaced with the Image. This will cause the image to repeat, which is usually undesirable.
- When false, the modifier is limited to faces with the Image as their Face Texture.

Projectors
: Up to ten projector objects are supported. Each face will choose the closest and aligned projector with its surface normal. Projections emit from the -Z axis (i.e. straight down a camera or lamp).
If the projector is a camera, the projection will adhere to its perspective/orthographic setting.

Objects
: Specify the projector Object

Aspect X/Y and Scale X/Y
: These allow simple manipulation of the image. Only apply when a camera is used as projector Object.

Usage

General

UV Project is great for making spotlights more diverse, and also for creating decals to break up repetition.

The modifier's Image property is not generally used: instead, a Texture mapped to the UV layer that the modifier targets is added to the object's Material. This allows you to prevent the image from repeating by setting *Texture → Image Mapping → Extension* to *Clip*.

Perspective Cameras

When using perspective cameras or spot lamps, you will likely want to enable the **UV Project** Material Option (available in the materials panel), This uses a different UV interpolation to prevent distortion.

Note: This option is not yet available for Cycles

UV Warp Modifier

Mode: Any mode

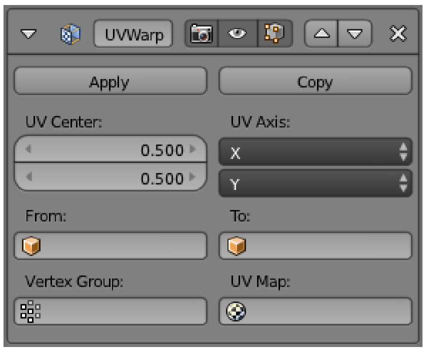
Panel: Modifiers (Generate)

File:Uvwarp.jpg
Projecting the Blender
logo onto Suzanne.

UV Warp uses 2 objects to define a transformation which is applied to the UV coordinates.

Download an example

Options



UV Center
: The center point of the UV map to use when applying scale or rotation. With (0, 0) bottom left and (1, 1) top right. Defaults to (0.5, 0.5).

UV Axis
The axis to use when mapping the 3D coordinates into 2D.

From/To
: The two objects used to define the transformation, when these objects overlap eachother - there will be no warp value, only the difference in transformation is used, note that bones can be used for armature objects.

Vertex Group
The vertex group can be used to scale the influence of the transformation per-vertex.

UV Map
: Which UV map to modify. Defaults to the active rendering layer.

Usage

The UV Warp modifiers main usage is to give you direct control over UV's as you do with objects and bones, so you can directly rotate, scale and translate existing UV coordinates using objects and bones.

WeightVGroup Modifiers

Mode: Any mode

Panel: Modifiers (Modifiers properties)

Description

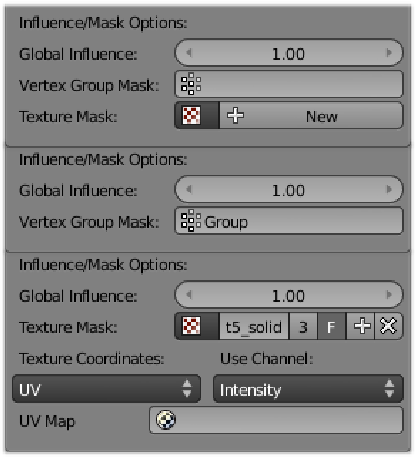
The WeightVGroup modifiers work on a vertex group of the affected object, by modifying its weights and/or which vertices belong to this group.

Those modifiers do implicit clamping of weight values in the standard `[0.0, 1.0]` range. So all values below **0.0**/above **1.0** will be lost!

There are currently three WeightVGroup modifiers:

- Vertex Weight Edit.
- Vertex Weight Mix.
- Vertex Weight Proximity.

Common Settings



The influence/masking part of WeightVGroup modifiers.

The three WeightVGroup modifiers share a few settings, controlling their influence on the affected vertex group.

Global Influence
: The overall influence of the modifier (**0.0** will leave the vertex group’s weights untouched, **1.0** is standard influence).
: Note that influence only affects weights, adding/removing of vertices to/from vertex group is not prevented by setting this value to **0.0**!

Vertex Group Mask
: An additional vertex group, which weights will be pre-multiplied with the global influence value, for each vertex. If a vertex is not in the masking vertex group, its masking weight (and hence its influence) will be null.

Texture
: An additional texture, which values will be pre-multiplied with the global influence value, for each vertex. You can choose which channel of the texture to use as values.
: This is a standard texture ID control. When set, it reveals other settings:

  Texture Coordinates
  : How the texture is mapped to the mesh… You have four choices:

    - Local: use local vertices coordinates.
    - Global: use the vertices coordinates in the global space.
    - Object: use the vertices coordinates in another object’s space.
    - UV: use an UV layer’s coordinates.

  Use Channel
  : Which channel to use as weight factor source (intensity, RGB, HSV, alpha – the options are quite self-explanatory, I guess…).

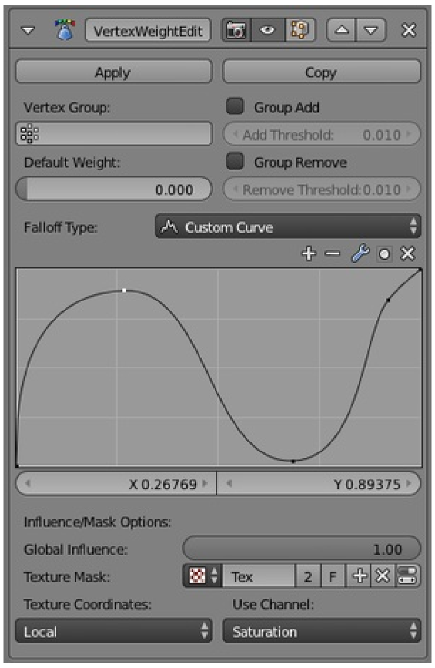
  Object
  : The object to be used as reference for Object mapping…

  UV Layer
  : The UV layer to be used for UV mapping…

Viewing Modified Weights

You will now view the modified weights in WeightPaint mode. This also implies that you'll have to disable the Vertex Weight modifiers if you want to see the original weights of the vertex group you are editing (provided it is affected by some modifier, obviously).

Vertex Weight Edit Modifier



The WeightVGEdit modifier panel.

This modifier is intended to edit the weights of one vertex group.

The general process is the following, for each vertex:

1. [Optional] It does the mapping, either through one of the predefined functions, or a custom mapping curve.
2. It applies the influence factor, and optionally the vertex group or texture mask (null values mean original weight, **1.0** ones mean fully mapped weight).
3. It applies back the weight to the vertex, and/or it might optionally remove the vertex from the group if its weight is below a given threshold, or add it if it's above a given threshold.

Options

Vertex Group
  The vertex group to affect.

Default Weight
  The default weight to assign to all vertices not in the given vertex group.

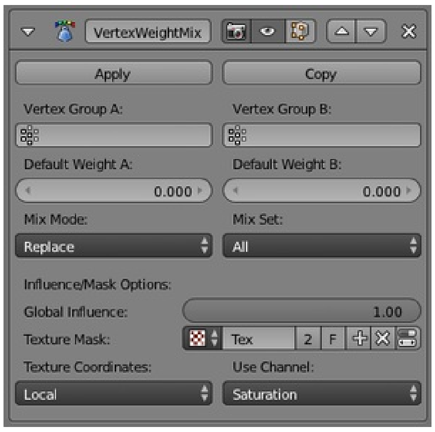
Falloff Type
  Type of mapping:

- Linear – No mapping.
- Custom Curve – Enables the the curve mapping. This shows up a curve control.
- Sharp, Smooth, Root and Sphere are classical mapping functions, from spikiest to roundest.
- Random – Fully randomizes the weights!
- Median Step – Creates binary weights (**0.0** or **1.0**), with **0.5** as cutting value.

Group Add
  Adds vertices with a final weight over Add Threshold to the vertex group.

Group Remove
  Removes vertices with a final weight below Rem Threshold from the vertex group.

Vertex Weight Mix Modifier



Vertex Group A:
Vertex Group B:
Default Weight A: 0.000
Default Weight B: 0.000
Mix Mode: Replace
Mix Set: All
Influence/Mask Options:
Global Influence: 1.00
Texture Mask: Tex 2 F
Texture Coordinates: Local
Use Channel: Saturation

The WeightVGMix modifier panel.

This modifier mixes a second vertex group (or a simple value) into the affected vertex group, using different operations.

It also has an option to choose which vertices to work on (all, only those of the first or second vertex group, etc.).

This implies that it *might* add vertices to the affected vertex group (it will never remove vertices, though); see below for details.

Options

Vertex Group A
: The vertex group to affect.

Default Weight A
: The default weight to assign to all vertices not in the given vertex group.

Vertex Group B
: The second vertex group to mix into the affected one. Leave it empty if you only want to mix in a simple value.

Default Weight B
: The default weight to assign to all vertices not in the given second vertex group.

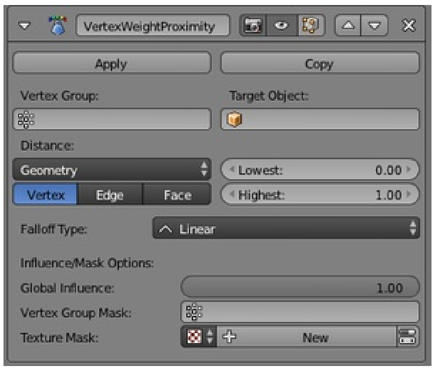
Mix Mode
: How the vertex group weights are affected by the other vertex group’s weights. You have seven choices:

- Replace weights just replaces affected weights by the second weights.
- Add to weights adds both values.
- Subtract from weights subtracts the second weights from the affected weights.
- Multiply weights multiplies both weights.
- Divide weights divides the affected weights by the second weights.
- Difference computes the difference between affected weights and second weights (it’s just the absolute value of the subtract operation).
- Average computes the average value of both weights.

Mix Set
: Which vertices to work on. You have five options:

- All vertices affects all vertices, disregarding the vertex groups content. *This option might add vertices to the affected vertex group.*
- Vertices from group A affects only vertices belonging to the affected vertex group.
- Vertices from group B affects only vertices belonging to the second vertex group. *This option might add vertices to the affected vertex group.*
- Vertices from one group affects only vertices belonging to at least one of the vertex groups. *This option might add vertices to the affected vertex group.*
- Vertices from both groups affects only vertices belonging to both vertex groups.

Vertex Weight Proximity Modifier



The WeightVGProximity modifier panel.

This modifier sets the weights of the given vertex group, based on the distance between the object (or its vertices), and another target object (or its geometry).

Options

Vertex Group
: The vertex group to affect.

Target Object
: The object from which to compute distances.

Proximity mode

- Object Distance will use the distance between the modified mesh object and the target object as weight for all vertices in the affected vertex group.
- Geometry Distance will use the distance between each vertex and the target object, or its geometry.

The Geometry Distance mode has three additional options, to use the target object’s geometry instead of its center location (if you enable more than one of them, the shortest computed distance will be used). If the target object has no geometry (e.g. an empty or camera one), it will silently fall back to the default Object Distance behavior.

Vertex
: This will set each vertex’s weight from its distance to the nearest vertex of the target object.

Edge
: This will set each vertex’s weight from its distance to the nearest edge of the target object.

Face
: This will set each vertex’s weight from its distance to the nearest face of the target object.

Lowest Dist
: Distance mapping to **0.0** weight. It can be above Highest Dist for reversed mapping effects.

Highest Dist
: Distance mapping to **1.0** weight. It can be below Lowest Dist for reversed mapping effects.

Falloff Type
: Some predefined mapping functions, see the Vertex Weight Edit part above.

Examples

Using Distance from a Target Object

As a first example, let’s dynamically control a Wave modifier with a modified vertex group.

Add a Grid mesh, with many vertices (e.g. a **100×100** vertices), and **10** BU side-length. Switch to Edit mode (⭾ Tab), and in the Object Data properties, Vertex Groups panel, add a vertex group. Assign to it all your mesh’s vertices (with e.g. a **1.0** weight). Go back to Object mode.

Then, go to the Modifiers properties, and add a Vertex Weight Proximity modifier. Set the mode to Object Distance. Select your vertex group, and the target object you want (here I used the lamp).

You will likely have to adjust the linear mapping of the weights produced by the Vertex Weight Proximity modifier. To do so, edit Lowest Dist and Highest Dist so that the first corresponds to the distance between your target object and the vertices you want to have lowest weight, and similarly with the second and highest weight…

Now add a Wave modifier, set it to your liking, and use the same vertex group to control it.

Animate your target object, making it move over the grid. As you can see, the waves are only visible around the reference object! Note that you can insert a Vertex Weight Edit modifier before the Wave one, and use its Custom Curve mapping to get larger/narrower “wave influence’s slopes”.

[video link]

The Blender file, `TEST_1` scene.

Using Distance from a Target Object’s Geometry

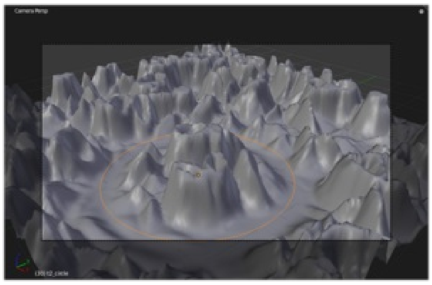
We’re going to illustrate this with a Displace modifier.

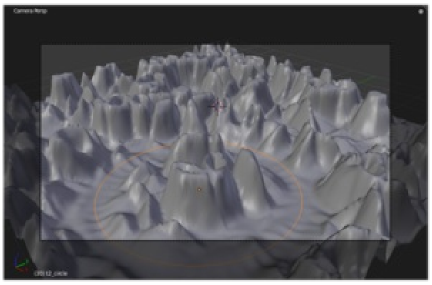
Add a **10×10** BU **100×100** vertices grid, and in Edit mode, add to it a vertex group containing all of its vertices, as above. You can even further sub-divide it with a first Subsurf modifier.

Now add a curve circle, and place it **0.25** BU above the grid. Scale it up a bit (e.g. **4.0**).

Back to the grid object, add to it a Vertex Weight Proximity modifier, in Geometry Distance mode. Enable Edge (if you use Vertex only, and your curve has a low U definition, you would get wavy patterns, see (*Wavy patterns*)).

Wavy patterns.


Distance from edges.

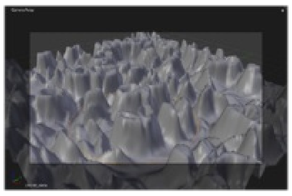

Distance from vertices.

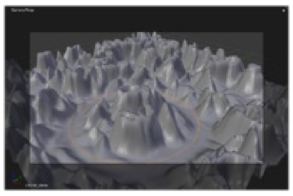
Set the Lowest Dist to **0.2**, and the Highest Dist to **2.0**, to map back the computed distances into the regular weight range.

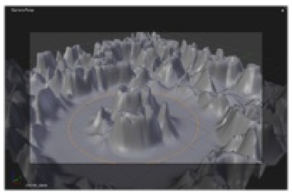
Add a third Displace modifier and affect it the texture you like. Now, we want the vertices of the grid nearest to the curve circle to remain undisplaced. As they will get weights near zero, this means that you have to set the Midlevel of the displace to **0.0**. Make it use our affected vertex group, and that’s it! Your nice mountains just shrink to a flat plane near the curve circle.

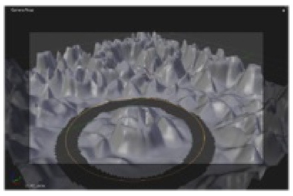
As in the previous example, you can insert a Vertex Weight Edit modifier before the Displace one, and play with the Custom Curve mapping to get a larger/narrower “valley”…

Curve Map variations.


Concave-type mapping curve.


No mapping curve (linear).


Convex-type mapping curve.


Vertices with a computed weight below **0.1** removed from the vertex group.

You can also add a fifth Mask modifier, and enable Vertex Weight Edit’s Group Remove option, with a Rem Threshold of **0.1**, to see the bottom of your valley disappear.

[video link]

The Blender file, `TEST_2` scene.

Using a Texture and the Mapping Curve

Here we are going to create a sort of strange alien wave (yes, another example with the Wave modifier… but it’s a highly visual one; it’s easy to see the vertex group effects on it…).

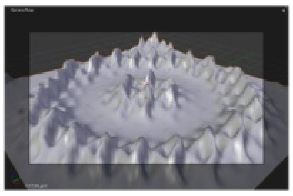
So as above, add a **100×100** grid. This time, add a vertex group, but without assigning any vertex to it – we’ll do this dynamically.

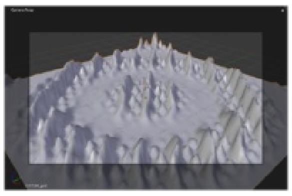
Add a first Vertex Weight Mix modifier, set the Vertex Group A field with a Default Weight A of **0.0**, and set Default Weight B to **1.0**. Leave the Mix Mode to Replace weights, and select All vertices as Mix Set. This way, all vertices are affected. As none are in the affected vertex group, they all have a default weight of **0.0**, which is replaced by the second default weight (**1.0**). And all those vertices are also added to the affected vertex group.

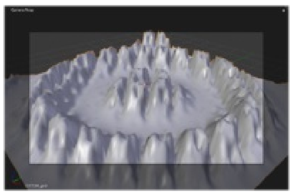
Now, select or create a masking texture – here I chose a default Magic one. The values of this texture will control how much of the “second weight” (**1.0**) replaces the “first weight” (**0.0**)… In other words, they are taken as weight values!

You can then select which texture coordinates and channel to use. Leave the mapping to the default Local option, and play with the various channels…

Texture channel variations.


Using intensity.


Using Red.

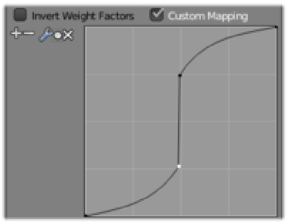

Using Saturation.

Don’t forget to add a Wave modifier, and select your vertex group in it!

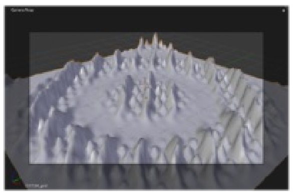
You can use the weights created this way directly, but if you want to play with the curve mapping, you must add the famous Vertex Weight Edit modifier, and enable its Custom Curve mapping.

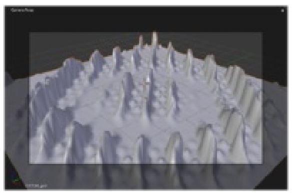
By default, it’s a one-to-one linear mapping – in other words, it does nothing! Change it to something like in (*A customized mapping curve*), which maps `[0.0, 0.5]` to `[0.0, 0.25]` and `[0.5, 1.0]` to `[0.75, 1.0]`, thus producing nearly only weights below **0.25**, and above **0.75**: this creates great “walls” in the waves…

Custom mapping curve.



A customized mapping curve.


Custom Mapping disabled.


Custom Mapping enabled.

[video link]

The Blender file, `TEST_4` scene.

See Also

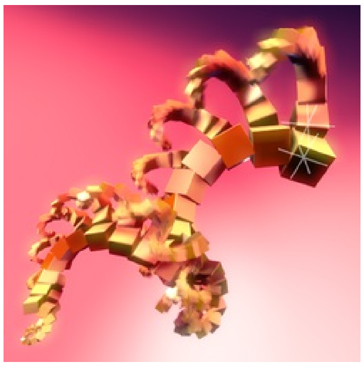
- The Development page.

Array Modifier

Mode: Object mode

Panel: Properties Window -> Context Button Modifiers

Description

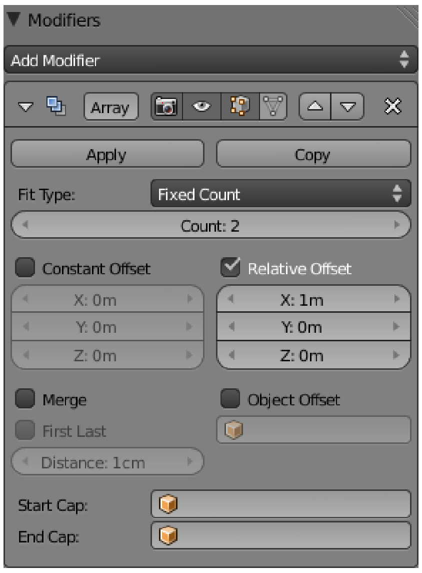

Multidimensional array animated with motion blur.

The Array modifier creates an array of copies of the base object, with each copy being offset from the previous one in a number of possible ways. Vertices in adjacent copies can be merged based on a merge distance, allowing smooth subsurf frameworks to be generated.

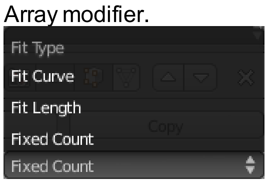
This modifier can be useful when combined with tilable meshes for quickly developing large scenes. It is also useful for creating complex repetitive shapes.

Multiple array modifiers may be active for an object at the same time, e.g. to create complex 3 dimensional constructs.

Options



Array modifier.



Fit Type menu.

Fit Type menu
: Controls how the length of the array is determined. There are three choices, activating respectively the display of the Curve, Length or Count setting:

  - Fit Curve – Generates enough copies to fit within the length of the curve object specified in Curve.
  - Fit Length – Generates enough copies to fit within the fixed length given by Length.
  - Fixed Count – Generates the number of copies specified in Count.

Curve
: The Curve object to use for Fit Curve.

Length
: The length to use for Fit Length.

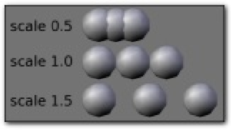
Count
: The number of duplicates to use for Fixed Count.

Notes

- Both Fit Curve and Fit Length use the local coordinate system size of the base object, which means that scaling the base object in Object mode will not change the number of copies generated by the Array modifier.
- Fit Length uses the local coordinate system length of the curve, which means that scaling the curve in Object mode will not change the number of copies generated by the Array modifier.
- Applying the scale with the Apply Scale button can be useful for each one.

Constant Offset, X, Y, Z
: Adds a constant translation component to the duplicate object's offset. X, Y and Z constant components can be specified.

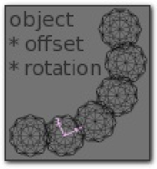
Relative Offset, X, Y, Z



Relative offset example.

Adds a translation equal to the object's bounding box size along each axis, multiplied by a scaling factor, to the offset. X, Y and Z scaling factors can be specified. See *Relative offset example*.

Object Offset

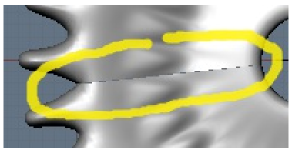


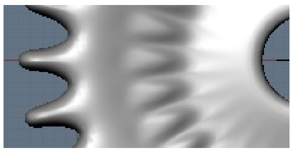
Object offset example.

Adds a transformation taken from an object (relative to the current object) to the offset. See *Object offset example*. It is good practice to use an Empty object centered or near to the initial object. E.g. by rotating this Empty a circle or helix of objects can be created.

Merge
: If enabled, vertices in each copy will be merged with vertices in the next copy that are within the given Distance.

First Last
: If enabled **and** Merge is enabled, vertices in the first copy will be merged with vertices in the last copy (this is useful for circular objects; see (*First Last merge example*)).


Subsurf discontinuity caused by not merging vertices between first and last copies (First Last off).


Subsurf discontinuity eliminated by merging vertices between first and last copies (First Last on).

First Last merge example.

Distance
: Controls the merge distance for Merge.

Start cap
: The mesh object to be used as a start cap. A single copy of this object will be placed at the "beginning" of the array – in fact, as if it was in position **-1**, i.e. one "array step" before the first "regular" array copy. Of course, if Merge is activated, and the Start cap is near enough to the first copy, they will be merged.

End cap
: The mesh object to be used as an end cap. A single copy of this object will be placed at the "end" of the array – in fact, as if it was in position **n+1**, i.e. one "array step" after the last "regular" array copy. And as Start cap, it can be merged with the last copy…

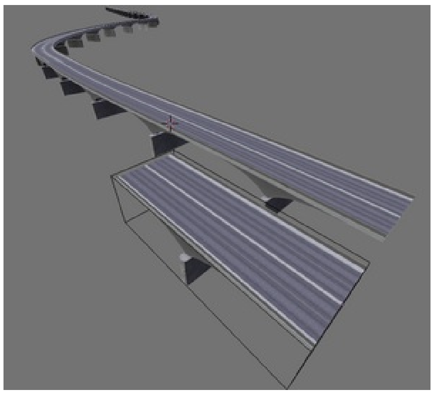
Hints

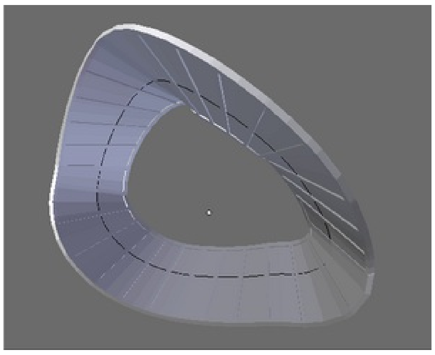
Offset Calculation

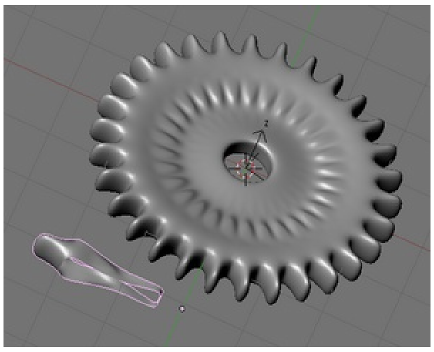
The transformation applied from one copy to the next is calculated as the sum of the three different components (Relative, Constant and Object), all of which can be enabled/disabled independently of the others. This allows, for example, a relative offset of **(1, 0, 0)** and a constant offset of **(0.1, 0, 0)**, giving an array of objects neatly spaced along the X axis with a constant **0.1BU** (Blender Units) between them, whatever the original object's size.

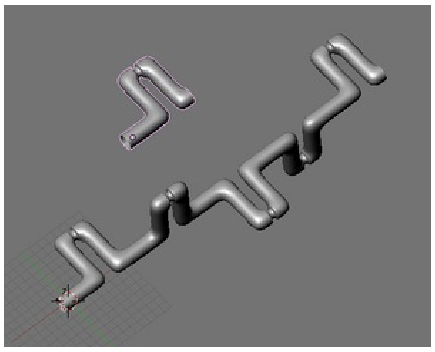
Examples

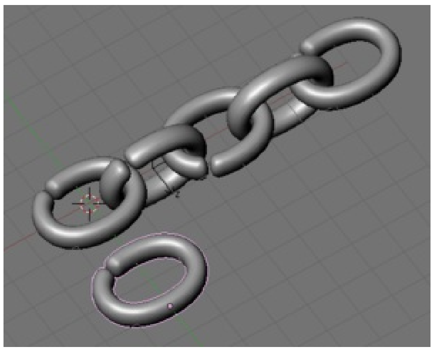
Mechanical


A bridge made from a tileable mesh.

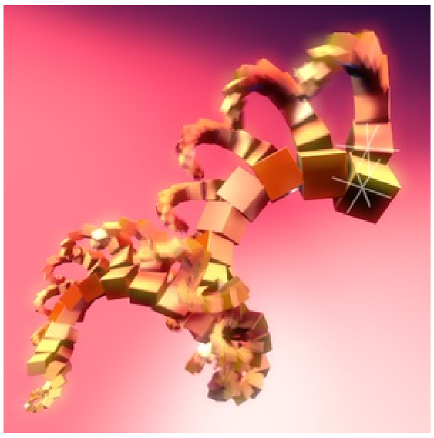

A track.
Sample blend file

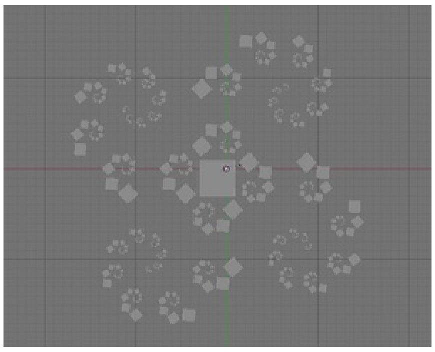

A cog created from a single segment.
Sample blend file


A crankshaft.
Sample blend file

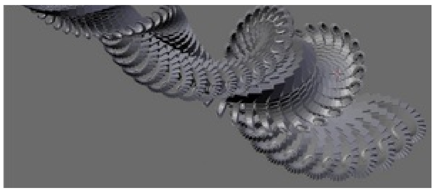

A chain created from a single link.
Sample blend file

Fractal


Multidimensional array animated with

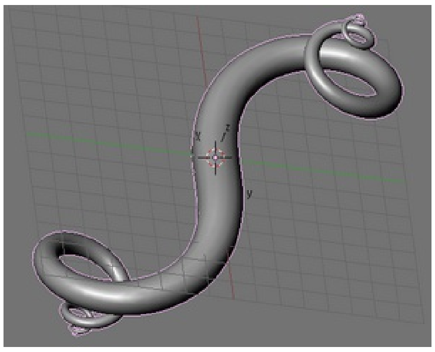

A fractal-like image created with multiple array modifiers applied to a cube.
Sample blend file

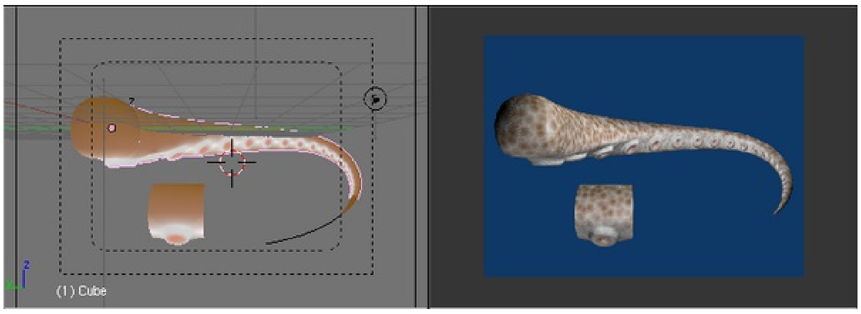
motion blur.


A fractal fern image created with 2 array modifiers and 1 mirror applied to a cube.

Organic

Subsurfed cube array with 1 object offset, 4 cubes and a high vertex merge setting to give the effect of skinning.


A double spiral created with two array modifiers and one subsurf modifier applied to a cube. As above, the vertex merge threshold is set very high to give the effect of skinning.
Sample blend file


A tentacle created with an Array modifier followed by a Curve modifier. The segment in the foreground is the base mesh for the tentacle; the tentacle is capped by two specially-modeled objects deformed by the same Curve object as the main part of the tentacle.
Sample blend file

Tutorials

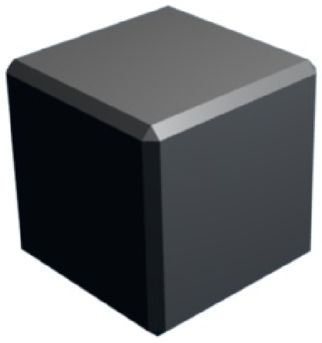
- Neal Hirsig's Array Modifier Screencast on Vimeo
- Creating A Double Helix With Modifiers

The 'Double Helix' tutorial explains the Array modifier. It is for an old Blender Version (2.44) but except for the keyboard shortcuts it is still valid.

Bevel Modifier

Mode: Object mode

Panel: Properties Window -> Context Button Modifiers


Default bevel.

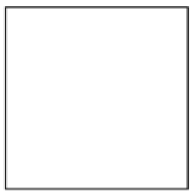
The Bevel modifier adds the ability to bevel the edges of the mesh it is applied to, allowing control of how and where the bevel is applied to the mesh.

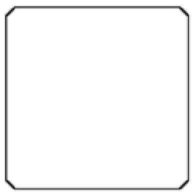
The Bevel modifier is a non-destructive alternative to the Bevel Operation in mesh editing mode.

The picture *Unbeveled square* shows a square which has unbeveled edges as the angles between the corners of the square are 90° (perpendicular). The picture *Beveled square* shows a square which has beveled corners.

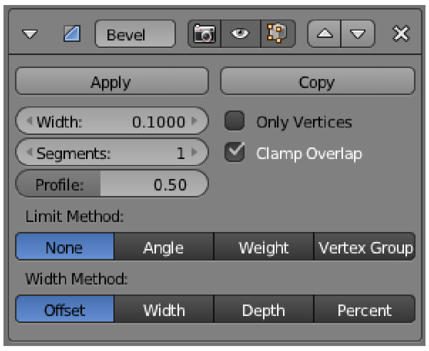
Although the two pictures show 2D squares, the Blender Bevel modifier can work on both 2D and 3D meshes of almost any shape, not just squares and cubes…

The picture *Default bevel* shows a Blender 3D cube with a bevel applied using just the default Bevel modifier settings.


Unbeveled square.


Beveled square.

Options

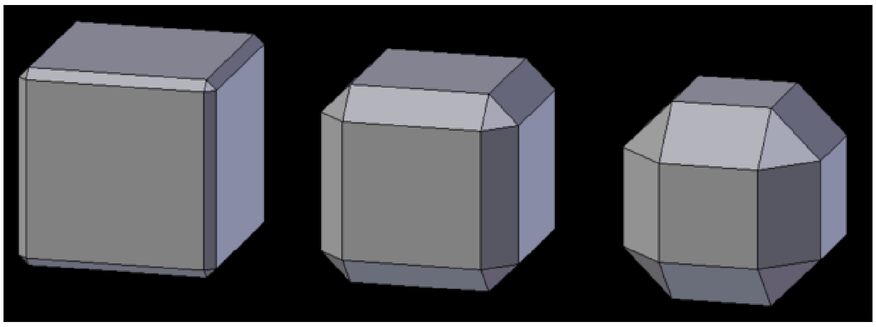


Bevel modifier panel.

Width

The Width numeric field controls the width/amount of the bevel applied to the base mesh. It can range from **0.0** (no bevel applied) to **1.0** (Blender Units). This value is in fact the “backing up” of the two new edges relatively to the original (beveled) one, along the two concerned faces.

Note
When using Metric or Imperial units the Width value has a unit. E.g. when 1 Blenderunit is 1m a useful value is between 0cm and 100cm. When it seems that decreasing the Width has no effect anymore check if the unit changed to m instead of cm.

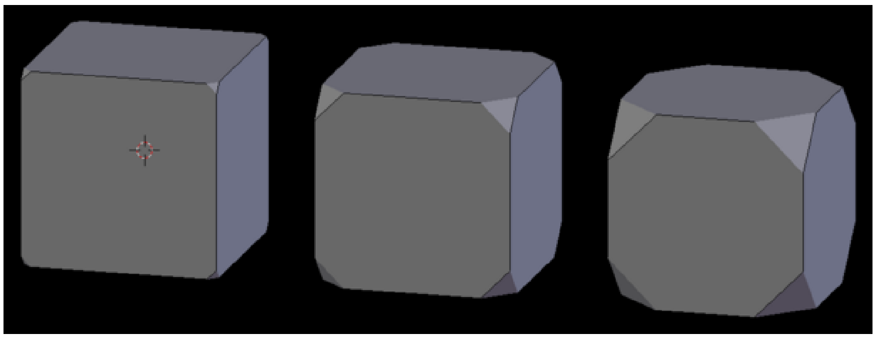

Three Cubes with **0.1**, **0.3** and **0.5** bevel Widths.

Segments

Set the number of bevel segments for round edges/verts.

Only Vertices

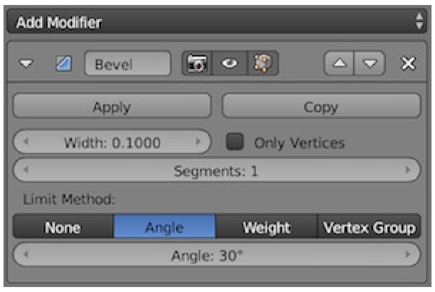
The Only Vertices button alters the way in which a bevel is applied to the mesh. When it is active, only the areas near vertices are beveled; the edges are left unbeveled.


Three cubes with **0.1'**, **0.3** and **0.5** bevel Widths, with Only Vertices option enabled.

Limit Method

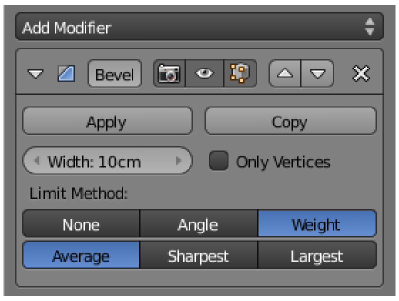
This section of the Bevel modifier is used to control where and when a bevel is applied to the underlying mesh. The first row of three buttons (mutually exclusive) controls the algorithm used, and might add some extra options.

None
: This button will apply the Bevel modifier to the whole underlying mesh, without any way to prevent the bevel on some edges/vertices.



Bevel modifier with the Angle limit enabled.

Angle
: This button will only bevel edges where faces make sharp angles. When selected, it displays the Angle numeric field, used to set the angle above which an edge will be beveled (it is in fact the complementary angle, i.e. `180°-(angle between faces)`). When the angle between meeting faces is less than the angle in the slider box, a bevel on those specific edges will not be applied. Similarly, when the angle between two edges is less than this limit, the vertex is not beveled.



Bevel modifier with Weight button active.

Weight
: Use bevel weights to determine how much bevel is applied; apply them separately in vert/edge select mode. See Here about adjusting bevel weights. The three options specify what edge weight to use for weighting a vertex.
: Average

    Uses the average bevel weight at the vertex

: Sharpest

    Uses the smallest bevel weight at the vertex

Largest

Uses the largest bevel weight at the vertex

Vertex Group

Use a vertex group to determine which parts of the mesh get beveled.

Boolean Modifier

Mode: Any 3d View Mode, can only be applied when in Object Mode.

Panel: Modifiers

Introduction

The Boolean Modifier performs operations on meshes that are otherwise too complex to achieve with few steps by editing meshes manually, meaning you can achieve good results with little to no effort to make mesh operations like Unions, Differences and Intersections. The Boolean modifier uses one of three Boolean operations (Difference (negation), Union (conjunction), and Intersect (disjunction)) to create a single compound object out of two Mesh objects.

- See Fig. 1 - The Union, Intersection and Difference between a Cube and a UV Sphere, with the modifier applied to the Sphere and using the cube as target (Detail, Union is using Ngons).

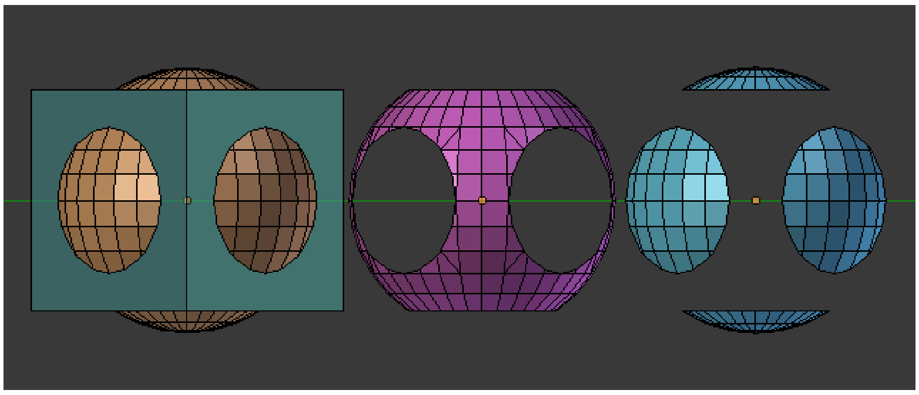


Fig. 1 - The Union, Intersection and Difference between a Cube and a UV Sphere, with the modifier applied to the Sphere and using the cube as target.

Description

The Boolean Modifier applies only for Mesh objects.

It performs one of the three Boolean Operations for the faces of open or closed volumes that creates a complete topology in the faces it's being used. This means that this modifier will only work properly for the intersection of faces of the two meshes that will result in another closed loop of edges (filled with faces), creating a new resulting face topology.

The Boolean modifier is non-destructive for the target; it uses the topology of the target to make the calculations, but you will still have the target in the scene. In normal conditions, using face normals pointed outside, when you apply the Boolean modifier operation, the modified mesh will receive changes in topology, and you will have to move the target to see the resulting mesh. The only exception is when you are using inverted normals; in this case, depending on the calculations, you will also change the topology of the target. You can see one example of a target being modified in the Materials section in this page, see Fig. 7 and 8.

You can add this Modifier in any of the Blender Modes, including some other Modes of different Blender screens, but the results of the mesh operation will only be shown in the Object Mode of the 3D View Window.

- The Boolean modifier works with open or closed volumes.
- The Boolean modifier doesn't work on edges without faces.
- The target topology determines the new topology of the modified mesh.
- The Face normals are taken into account for the calculations.
- Whether faces are marked for smooth or flat for shading doesn't affect the calculations of this modifier. (See Fig. 28 and 29.)
- The line at which this modifier is calculated is delimited by the first tangential contact between faces of the modified mesh and target.

This is a dynamic real-time modifier!

If you have marked your Objects to show the Edges (in Properties Window, Object Properties context, Display Tab, click *Wire*), you will see the Edge creation process while you're moving your Objects, but, depending on your mesh topology, you can also enable X-Ray and Transparency and see the topology being created in real time!

Usage

Using the default Blender install, with the desired mesh Object selected, go to the Properties Window which is located at the right of your Blender Screen, below the Outliner. Click on the Modifiers Context, which is represented by a wrench (see Fig. 2 - The Boolean Modifier; the wrench is highlighted in blue). Then, click on the Add Modifier Button and Blender will show you a list of all of the available Modifiers. The Boolean Modifier is located on the third row in the Generate Column.

You can also click on the *Add Modifier* Button and use N to add the Boolean Modifier, or use Blender search with the shortcut Space and type "Add Modifier" , click on *Add Modifier* and press N.

When you add the Boolean Modifier for an Object, Blender will need a second Object to perform the operation. You can use open or closed Meshes, as long as they have faces for calculations.

You can add one or more modifiers of this type for an Object but you can only apply one operation for the Boolean Modifier at a time.

Options

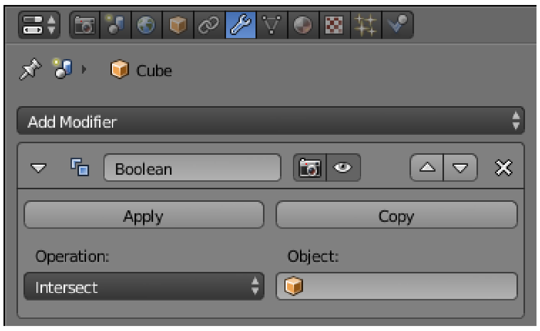


Fig. 2 - The Boolean Modifier

Input Box

In this Input Box you can give your Modifier a name. Blender default is *Boolean*.

- The Camera Button toggles the Modifier result to be visible during rendering, and the Eye toggles the Modifier result to be visible in the real-time session (with the effect shown only in Object Mode of the 3D View Window.)
- The Arrows let the user define the position of the Modifier in the modifiers stack when there are more modifiers applied to the object.
- The **X** is used to remove the modifier from the object.

Apply

This Button applies the operation to the modified mesh and only works in Object Mode of the 3D View Window. If you click on this Button in Edit Mode, Blender will present you with the standard message for modifiers, `Modifiers Cannot be applied in Edit Mode`.

Copy

Clicking in this Button will make Blender copy the Modifier, giving it a dot and a numeric suffix using three digits with a counter starting from 001 (e.g. Boolean.001).

Operations

Operation:

Difference

The modified mesh is subtracted from the target mesh.

- If the target Mesh uses inverted normals, Blender will Intersect the modified mesh.
- If the modified Mesh uses inverted normals, Blender will add both meshes (Union).
- If both Meshes use inverted normals, Blender will Intersect the target Mesh.

Union

The target mesh is added to the modified mesh.

- If the target Mesh uses inverted normals, Blender will Intersect the target Mesh.
- If the modified Mesh uses inverted normals, Blender will subtract the target Mesh.
- If both Meshes use inverted normals, Blender will Intersect the modified Mesh.

Intersect

The target mesh is subtracted from the modified mesh.

- If the target Mesh uses inverted normals, Blender will subtract the target Mesh.
- If the modified Mesh uses inverted normals, Blender will intersect the target Mesh.
- If both Meshes use inverted normals, Blender will add both meshes (Union).

Object

The name of the target object. Must be a mesh.

Materials

The Boolean Modifier preserves the Materials of the participant Meshes, including their basic textures and mappings, and the modified mesh will receive its first active material index assigned to its new topology (the first active material).

The only exception is the difference operation when the normals of the target and modified mesh are inverted (Fig 7 and 8). In this case, Blender will project the textures in an inverted direction over the target using the center contact of the meshes as a pivot and the resulting mesh will have the modified mesh subtracted from the target. For complex target meshes in some particular cases, you may have to reassign materials to faces because Blender will use the possible projection, and this may result in a sub-optimal texture assignment.

Below, some examples are shown to exemplify how materials work with the Boolean modifier; we took the cube as the modified mesh, and the icosphere as the target with one material (white). We added **4** different indexes to one of the faces of the cube, leaving another basic material in the other faces. Fig. 3 - Cube with Multi-Material Mesh (modified) and Icosphere (target) with basic material - shows how the Boolean modifier interacts with the materials. Figs. 4, 5 and 6 show three different Boolean operations applied to the modified mesh. The meshes used have normals pointed outwards (Normal meshes). See their captions for more information.

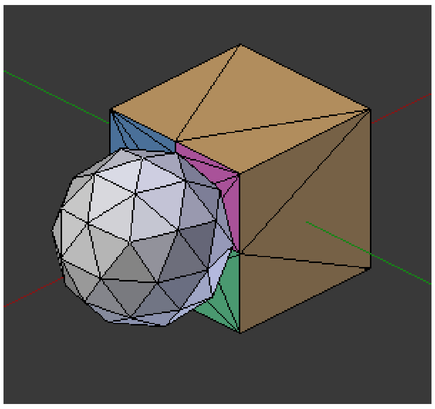


Fig. 3 - Cube with Multi-Material Mesh (modified) and Icosphere (target) with basic Material

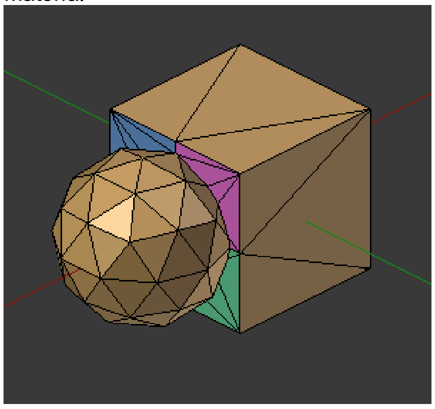


Fig. 4 - Union - The first active Material of the Cube is added to the new topology; other materials remain in the old topology

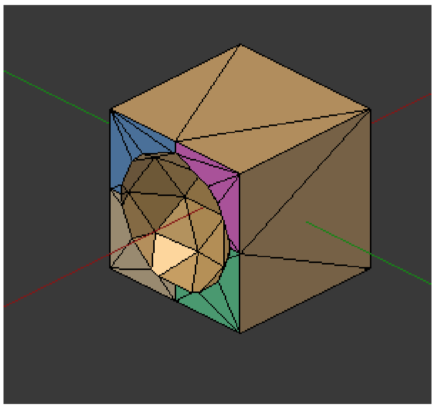


Fig. 5 - Difference - The Icosphere was subtracted from the Cube; the new topology has received the first active Material of the Cube

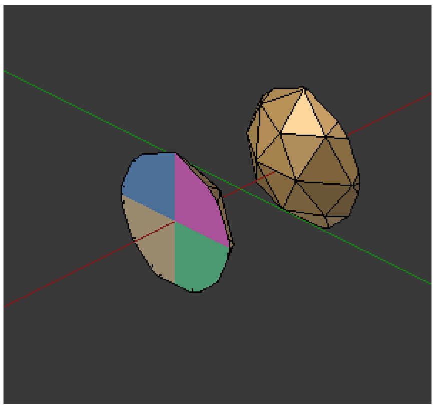


Fig. 6 - Intersect - The resulting Mesh was copied and rotated 180° - You can see the first active material of the cube in the back face (new topology); the front face received the 4 basic materials of the cube.

- In our last examples (Figs. 7 and 8) of how the Boolean modifier works with Materials, we have inverted normals for both the target (Icosphere) and modified mesh (Cube). As we said before, this is an exception rather than the rule. As you can see, the target received the materials of the modified mesh.

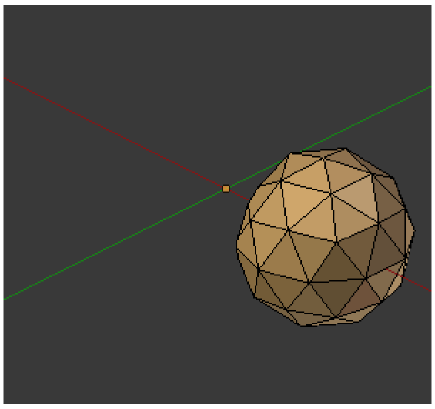


Fig. 7 - Front of the target with the modified mesh materials

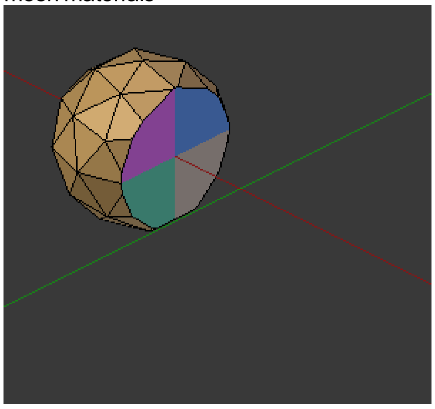


Fig. 8 - Back of the target with the modified mesh materials

UV Mappings

When you map UV Images to your target, Blender will add a map for each of the faces of the target. When you apply the Boolean modifier, Blender will follow the UV maps already assigned to the faces of the target topology that will be the result of the operation on the modified mesh. Blender will also use the same image mapped to the target faces in the modified mesh. But be aware that depending on your UV scheme (the way you have assigned textures to the faces during the UV unwrap), and the complexity of your meshes, the maps may not result in perfectly mapped UVs!

Below we have four Images, a UV sphere mapped with a test grid tinted blue and the other face tinted in purple, one face of the cube tinted in a light orange and the other faces using the normal test grid. Fig. 9 shows the operation at the start (difference), and on the right (Fig. 10), the resulting mesh. In Figs. 11 and 12 we show the unwrap in the Blender UV/Image Editor Window.

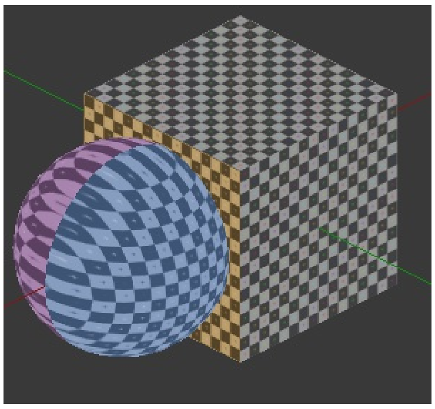


Fig. 9 - A UV Sphere and a Cube with different UV Maps

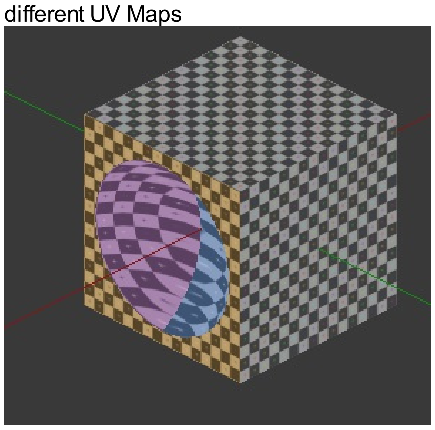


Fig. 10 - Difference operation applied

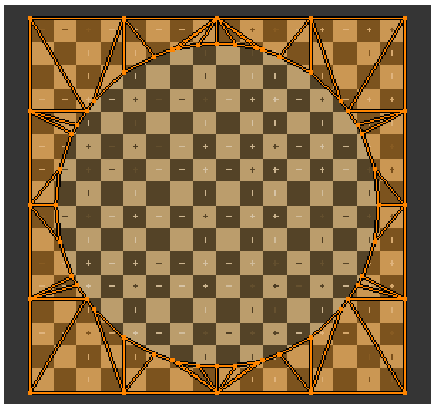


Fig. 11 - Faces of the modified mesh mapped

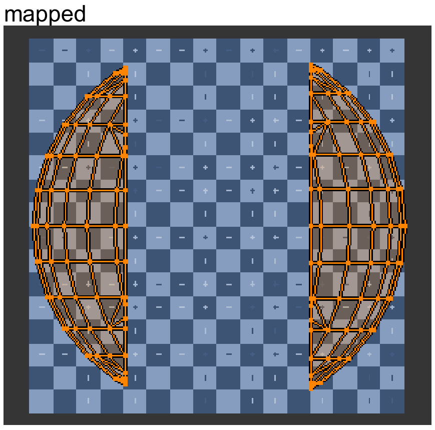


Fig. 12 - New topology mapped and UV faces assigned; we have another image assigned to the purple tinted faces.

Other Modifiers

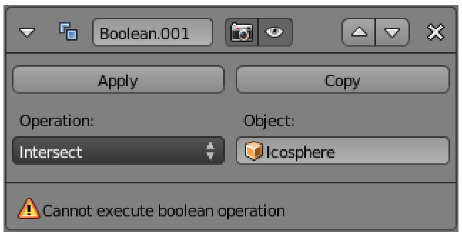


Fig. 13 - Boolean Modifier with error message

The Boolean Modifier calculation is performed using the target modified mesh topology and dimensions. Other modifiers added to the modified mesh are bypassed. It means that if a target is using another modifier, like subsurf, the resulting topology for the modified mesh will take into account the subsurf of the target; but for the modified mesh, the basic topology is used anyway (see examples).

If you add subsurf to the modified mesh with a Boolean modifier, Blender will visually add the subsurf for the modified mesh, but not for its calculations; it will only take into account its basic mesh topology. If you want to have a subsurf added to the modified mesh, you have to apply the subsurf to the Boolean modified mesh before applying the Boolean operation.

The Boolean Modifier can be added together with other modifiers in the modified mesh, but depending on the modifier, the calculations can't be done and/or the modifier cannot execute. When the modifier cannot execute, it will show the message `"Cannot execute boolean operation"` (see Fig. 13), and when the modifier cannot be applied to the mesh, Blender will show the message `"Modifier is disabled, Skipping Apply."`. In this case, you either have to remove some modifiers or apply the necessary ones.

The most common case is when you add or copy a Boolean modifier to use the modified mesh in conjunction with another target later; Blender will place the warning in the subsequent Boolean modifiers in the stack depending on the operation, because you may be creating concurrent Boolean operations for the same modified mesh, which in most cases is impossible to execute depending on the chosen target. In this case, you can apply the first Boolean modifier of the stack for the target and then use the other Boolean modifier(s) in the stack for subsequent operations.

Also, if some other modifiers are placed above this modifier and you click on Apply, Blender will warn you with the message `"Applied Modifier was not first, results may not be as expected"` . The best usage scenario for this modifier is to prepare your modified mesh and target to work with the Boolean modifier.

When the Boolean modifier is the first of the stack and is applied, the other Modifiers will act over the resulting meshes using the resulting topology and will remain in the modifiers stack.

Below are two images: one with the subsurf added to the target (Fig. 14), and another with the resulting topology (Fig. 15).

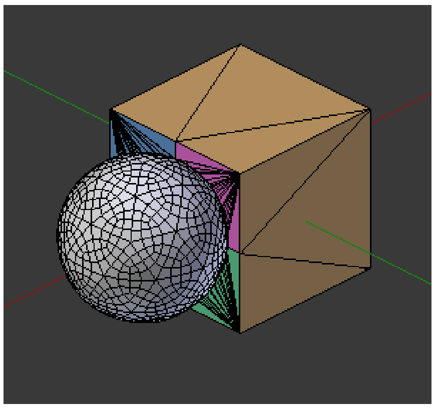


Fig. 14 - The Subsurf is only added to the target (Icosphere), not applied

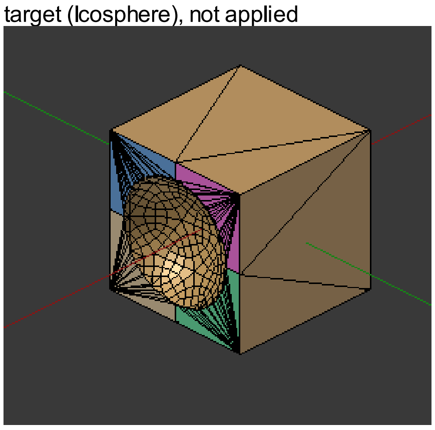


Fig. 15 - The resulting topology. The Subsurf added to the target was taken into account

- As you can see, the added (not applied) subsurf to the target was taken into consideration. The topology of the Icosphere with subsurf (Level 2) was completely transferred to the modified mesh.

The target topology determines the resulting topology

The target topology determines the results of the Boolean Modifier operation. It means that any modifier added to the target which modifies its topology will affect the resulting mesh of the operation.

Animation

The Boolean Modifier is a generating modifier, but its normal behavior is to be applied to static meshes. You can animate the target, the modified mesh or both, but the effects will only be visible when you render the edges of the modified mesh and the target to the final image or using recorded OpenGL animations.

Concurrent Operations

For the modified meshes, you can only apply one operation at a time, but you can use the same target for other modified meshes and use modified meshes as a target for other meshes as well. Also, you can copy or add the same modifier to the modifiers stack as many times as you wish to suit the number of operations you need, but be aware that if you choose concurrent targets which are, at the same time, modified meshes pointing to each other, you can cause Blender to crash with closed loops!

Hints

Be aware that other modifiers and their stack position could cause this modifier to fail in certain circumstances. Also, if you make two meshes act as a target for each other (in fact, creating a closed loop using concurrent operations), you can cause Blender to stop responding or crash.

The best usage scenario for sequential operations

The best way to work with this modifier when you need to make lots of sequential operations of the same modifier is to define the target at the time you need to apply the changes to the topology.

Face Normals

When using the Boolean Modifier, Blender will use the face normal directions to calculate the three Boolean operations. The direction of the normals will define the result of the three available calculations (see Operations in this page); when one of the participants uses a set of inverted normals, you're in fact multiplying the operation by **-1** and inverting the calculation order. You can, at any time, select your modified mesh, enter Edit Mode and flip the normals to change the behavior of the Boolean modifier. See Tips for fixing Normals in this page.

Blender also cannot perform any optimal Boolean operation when one or more of the mesh Normals of the participants that are touching has outwards/inwards normals mixed.

This means you can use the normals of the meshes pointed completely towards the inside or outside of your participants in the operation, but you cannot mix normals pointed inwards and outwards for the faces of the topology used for calculations. In this case, Blender will enable the modifier and you may apply the modifier, but with bad to no effects. We made some examples with a cube and an icosphere showing the results.

- See Fig. 16 and 17 - All face normals are pointing outwards (Normal meshes).

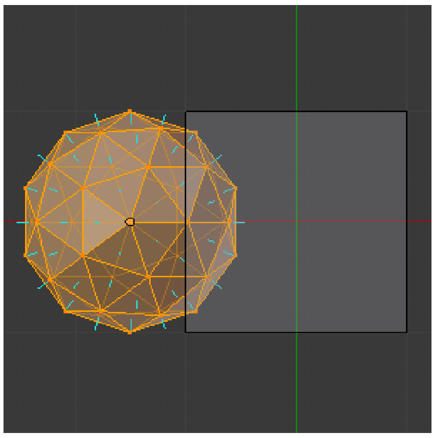


Fig. 16 - Faces with normals pointing outwards

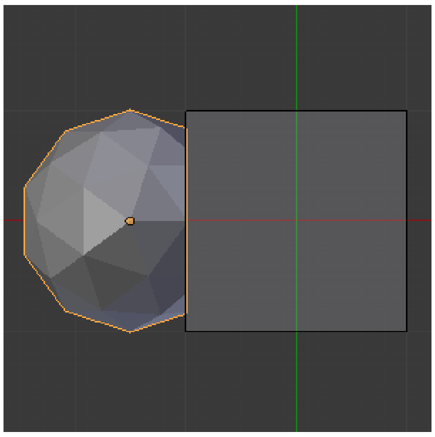


Fig. 17 - Normal Boolean modifier operation (Difference operation)

- See Fig. 18 and 19 - All face normals are pointing inwards (Meshes with inverted normals)

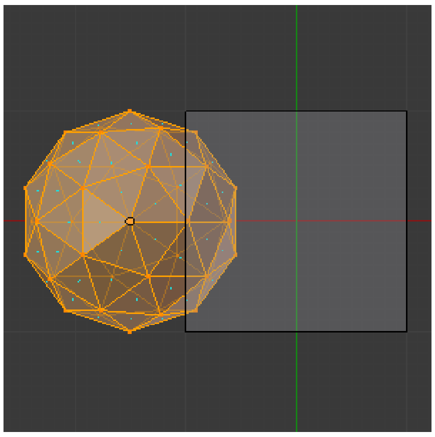


Fig. 18 - Faces with normals pointing inwards

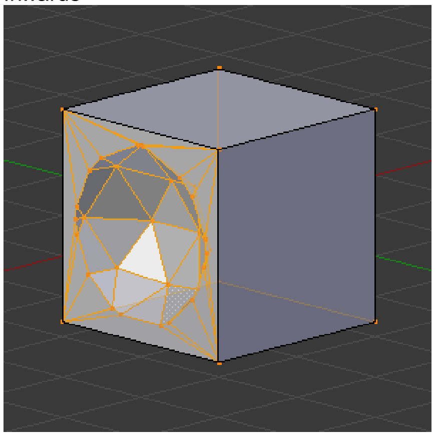


Fig. 19 - Normal Boolean modifier operation (Intersection operation)

- Now, let's see what happens when the normal directions are mixed for one of the participants in the Boolean Modifier operation. In Fig. 20 - Face normals mixed, pointed to different directions and 21 - Resulting operation, you can see that the modifier has bad effects when applied, leaving faces opened:

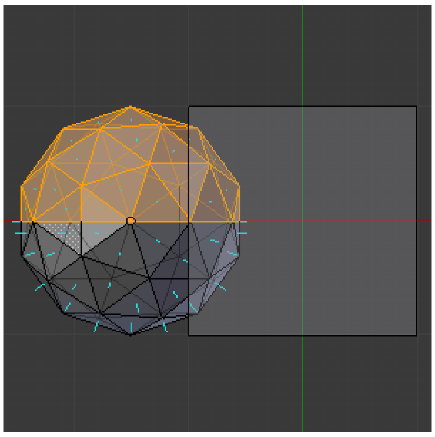


Fig. 20 - Face normals mixed, pointed to different directions

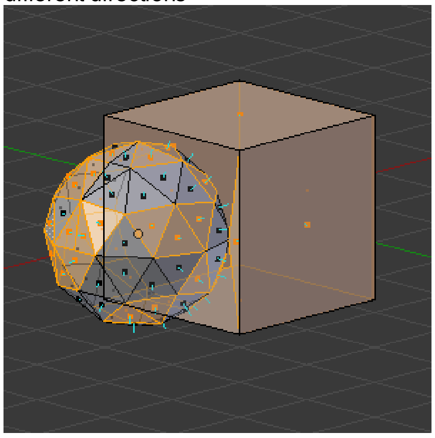
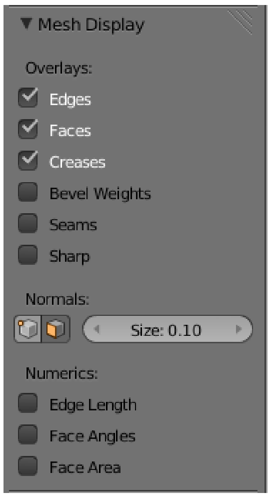


Fig. 21 - Resulting operation, Modifier has bad effect when applied, leaving faces opened

As you can see, the normal directions can be pointing to any of the Mesh sides, but can't be mixed in opposite directions for the faces of the participants. The Library can't determine properly what's positive and negative for the operation, so the results will be bad or you will have no effect when using the Boolean Modifier operation.

Tip for Fixing Mixed Normals

- You can fix mixed normals by recalculating them outside or inside; here we also give you a small hint on how to do this prior to Boolean Modifier usage:


▼ Mesh Display

Overlays:

- ✓ Edges
- ✓ Faces
- ✓ Creases
- Bevel Weights
- Seams
- Sharp

Normals:

Size: 0.10

Numerics:

- Edge Length
- Face Angles
- Face Area

Fig. 22 - Mesh Display in the Transform Panel

To show the normals of the faces, you can open the Transform Panel, find the Mesh display tab, and click on the small cube without the orange dot. (See Fig. 22 - Mesh Display in the Transform Panel.) You can also change the height of the axis that points the direction of the normal. The default is '*0.10*.

When some normal directions are mixed pointing inwards and outwards, you can recalculate them to the inside using ⇧ ShiftCtrlN and to outside using CtrlN. If the normals still get mixed due to Mesh complexities, you can change to Face selection Mode while in Edit Mode using Ctrl⇆ Tab and choosing *Face Mode*. Then select the faces that are pointing in the wrong direction using ⇧ Shift RMB and use the *Mesh* Menu entry in the Header of the 3D View Window, go to *Normals* and choose *Flip Normals*. (See Fig. 16 - Recalculate and Flip Normals in Mesh Menu Entry - 3D View.)

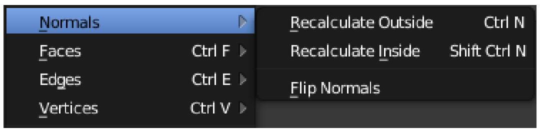


Fig. 23 - Recalculate and Flip Normals in Mesh Menu Entry - 3D View

Empty or Duplicated Faces

This modifier doesn't work when the modified and/or the target mesh uses empty faces in the topology used for calculations. If the modifier faces a situation where you have empty faces mixed with normal faces, the modifier will try, as much as possible, to connect the faces and apply the operation. For situations where you have two concurrent faces at the same position, the modifier will operate on the target mesh using both faces, but the resulting normals will get messed. To avoid duplicated faces, you can remove doubles for the vertices before recalculating the normals outside or inside. The button for remove doubles is located in the Mesh Tools Panel in the 3D View Window, while in Edit Mode.

The best usage scenario for this modifier is when you have clean meshes with faces pointing clearly to a direction (inwards/outwards)

Below we show an example of meshes with open faces mixed with normal faces being used to create a new topology. In this example, a difference between the cube and the icosphere is applied, but Blender connected a copy of the icosphere to the Cube mesh, trying to apply what was possible.

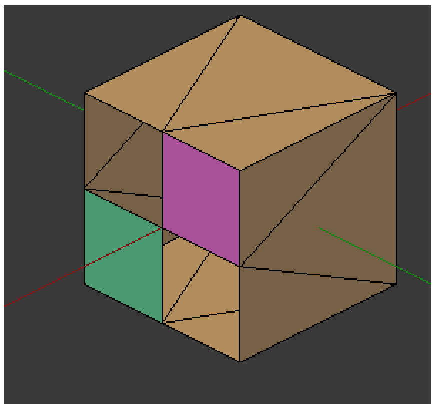


Fig. 24 - Mesh with two empty faces mixed with normal faces

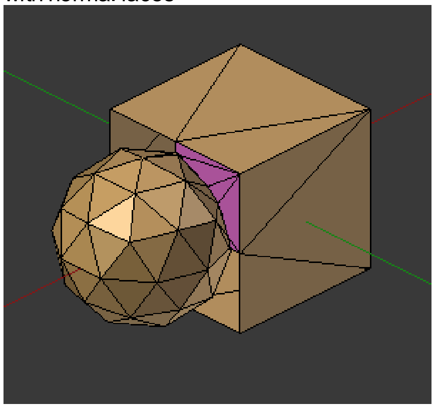


Fig. 25 - Result of a difference operation applied - Blender connected what was possible.

Open Volumes

The Boolean modifier permits you to use open meshes or non-closed volumes (not open faces).

When using open meshes or non-closed volumes, the Boolean modifier won't perform any operation in faces that don't create a new topology filled with faces using the faces of the target.

- See Fig. 26 and Fig. 27 - Resulting operation using two non-closed volumes with faces forming a new topology.

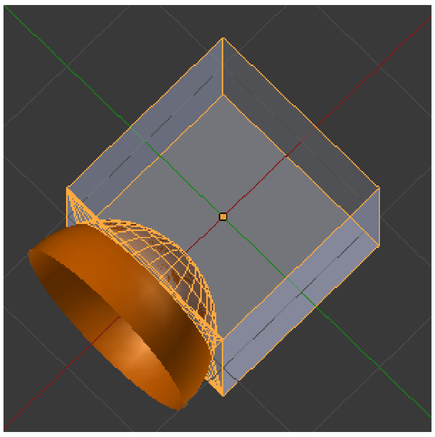


Fig. 26 - Non-closed volumes forming a new topology

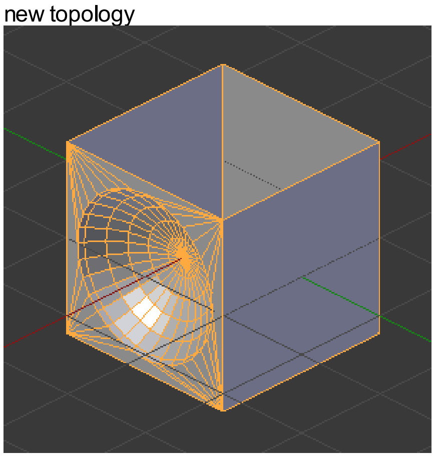


Fig. 27 - Resulting operation using two open volumes performing a new closed topology

- Now, let's see what happens when we use meshes that are partially open, incomplete, or meshes that aren't forming a new topology.

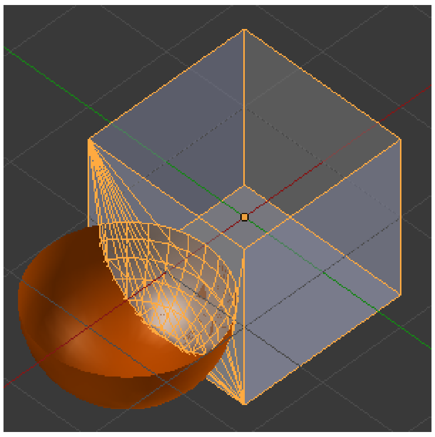


Fig. 28 - Open volumes that aren't forming a new topology.

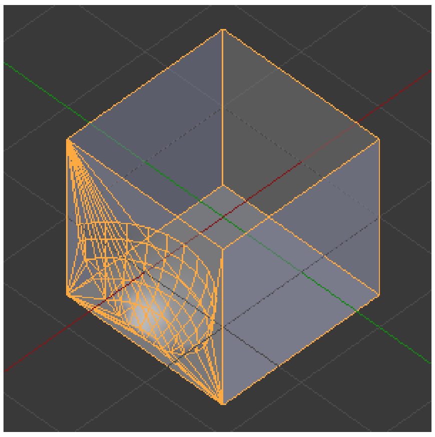


Fig. 29 - Resulting operation using two open volumes that aren't forming a new topology.

As you can see in Fig. 28, the faces of one participant in the Boolean operation gives incomplete information to the modifier; the result is shown in Fig. 29 - Resulting operation using two open volumes that aren't forming a new topology. The resulting edges get messy and there is not enough information to create faces for the resulting Mesh. This example uses a smooth shaded UVsphere cut in half. As explained before, the shading (smooth/flat) doesn't affect the calculations of the modifier.

History

Since version 2.62, Blender uses a new Library, the Carve library, which should give much improved results. This library is more stable and faster, resolving old well-known limitations of our previous library.

The general workflow and options available in the user interface are unchanged; usually the modifier will simply run faster and produce a better output mesh. However there are also some changes in behavior. In particular, Carve will perform Boolean operations only if the intersection of two meshes is a closed loop of edges.

release Notes and Development Page: Boolean Modifier

Useful Links

- Carve Developement Home - GPLv2 C++ source at Google Code
- Carve library - Homepage for the Carve Library project.
- Sculpt Tools - Link for a Blender Addon - This addon uses another approach to use the Boolean operations, when you select two or more objects, the active one becomes the modified mesh and all the others becomes a target. This addon will add the Boolean modifier and apply it to the meshes automatically.

Build Modifier

Mode: Object mode

Panel: Modifiers

The Build modifier causes the faces of the mesh object to appear, one after the other, over time. If the material of the mesh is a halo rather than a standard one, then the vertices of the mesh, not the faces, appear one after another.

By default, faces (or vertices) appear in the order in which they are stored in memory (by default, the order of creation). The face/vertex order can be altered in Edit mode by selecting Sort Faces from the Search Menu Space.

Options

Build modifier in action

Start
: The start frame of the building process.

Length
: The number of frames over which to rebuild the object.

Randomize
: Randomizes the order in which the faces are built.

Seed
: The random seed. Changing this value gives a different "random" order when "Randomize" is checked – this order being always the same for a given seed/object set.

Decimate Modifier

Mode: Object mode

Panel: Modifiers

Description

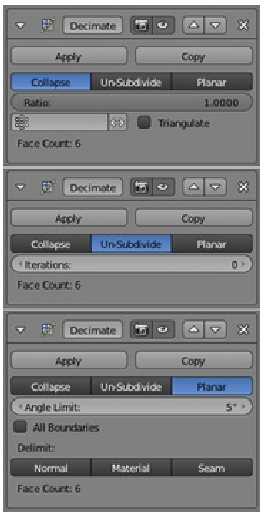
The Decimate modifier allows you to reduce the vertex/face count of a mesh with minimal shape changes. This is not applicable to meshes which have been created by modeling carefully and economically, where all vertices and faces are necessary to correctly define the shape, but if the mesh is the result of complex modeling, with proportional editing, successive refinements, possibly some conversions from SubSurfed to non-SubSurfed meshes, you might very well end up with meshes where lots of vertices are not really necessary.

The Decimate modifier is a quick and easy way of reducing the polygon count of a mesh non-destructively. This modifier demonstrates the advantages of a mesh modifier system because it shows how an operation which is normally permanent and destroys original mesh data, can be done interactively and safely using a modifier.

Unlike the majority of existing modifiers, the Decimate modifier does not allow you to visualize your changes in Edit mode.

Decimate only handles triangles, so each quadrilateral face is implicitly split into two triangles for decimation.

Options



Ratio: 1.0000
Face Count: 6
Iterations: 0
Face Count: 6
Angle Limit: 5°
All Boundaries
Face Count: 6

decimate modifier

Ratio
: The ratio of faces to keep after decimation, from **0.0** (0%, all faces have been completely removed) to **1.0** (100%, mesh is completely intact, except quads have been triangulated).
: As the percentage drops from **1.0** to **0.0**, the mesh becomes more and more decimated until it no longer visually looks like the original mesh.

Face Count (display only)
: This field shows the number of remaining faces as a result of applying the Decimate modifier.

Examples

Simple plane

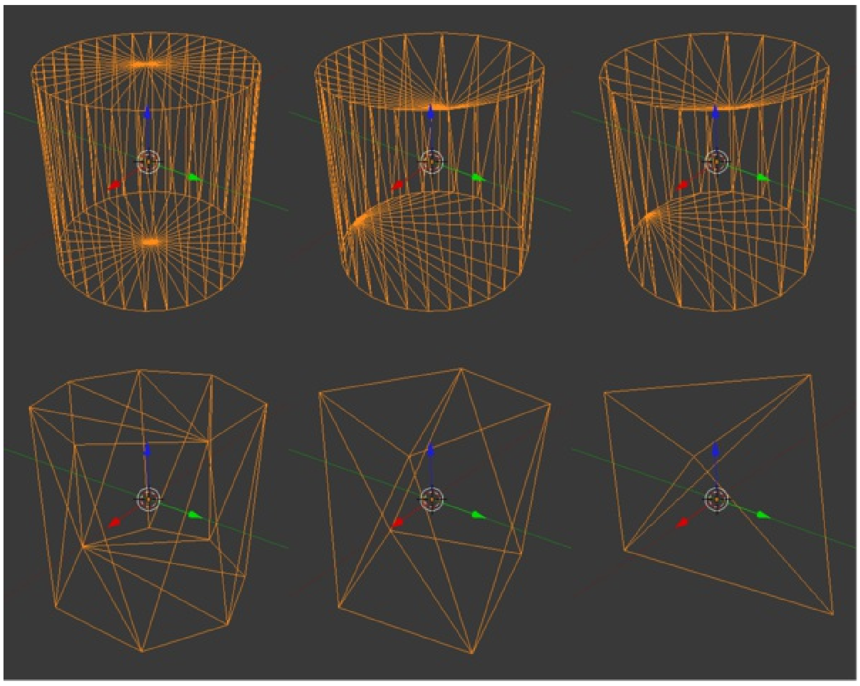
A simple example is a plane, and a 4x4 undeformed Grid object. Both render exactly the same, but the plane has one face and four vertices, while the grid has nine faces and sixteen vertices, hence lots of unneeded vertices and faces. The Decimate modifier allows you to eliminate these unneeded faces.

Decimated cylinder

We take a simple example of decimating a cylinder with the default of **32** segments. This will generate a cylinder with **96** faces. When the Decimate modifier is applied, the face count goes up! This is because the modifier converts all quadrangles (*quads*) into triangles (*tris*) which always increases the face count. Each quad decomposes into two triangles.

The main purpose of the Decimate modifier is to reduce mesh resources through a reduction of vertices and faces, but at the same time maintain the original shape of the object.

In the following picture, the percentage dropped in each successive image, from **100%** to **5%** (a ratio of **0.05**). Notice that the face count has gone from **128** to **88** at a ratio of **0.6** (**60%**) and yet the cylinder continues to look very much like a cylinder, and we discarded **40** unneeded faces.


1.0 (100%). Faces: 128; 0.8 (80%). Faces: 102; 0.6 (60%). Faces: 88
0.2 (20%). Faces: 24; 0.1 (10%). Faces: 12; 0.05 (5%). Faces: 6

As you can see, when the ratio reaches **0.1**, the cylinder looks more like a cube. And when it reaches **0.05**, it doesn’t even look like a cube!

Once you have reached the face count and appearance you were looking for, you can Apply the modifier. If you want to convert many of the tris back to quads to reduce mesh resources further, you can switch to Edit mode, select all vertices (A), and hit AltJ.

EdgeSplit Modifier

Mode: Any mode

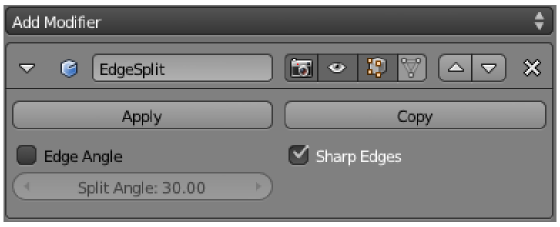
Panel: Properties Window -> Context Button Modifiers

Description

The EdgeSplit modifier splits edges within a mesh. The edges to split can be determined from the edge angle (i.e. angle between faces forming this edge), and/or edges marked as sharp.

Splitting an edge affects vertex normal generation at that edge, making the edge appear sharp. Hence, this modifier can be used to achieve the same effect as the Autosmooth button, making edges appear sharp when their angle is above a certain threshold. It can also be used for manual control of the smoothing process, where the user defines which edges should appear smooth or sharp (see Mesh Smoothing for other ways to do this). If desired, both modes can be active at once. The output of the EdgeSplit modifier is available to export scripts, making it quite useful for creators of game content.

Options



Edge Angle [unchecked]
Sharp Edges [✓]
Split Angle: 30.00

EdgeSplit modifier.

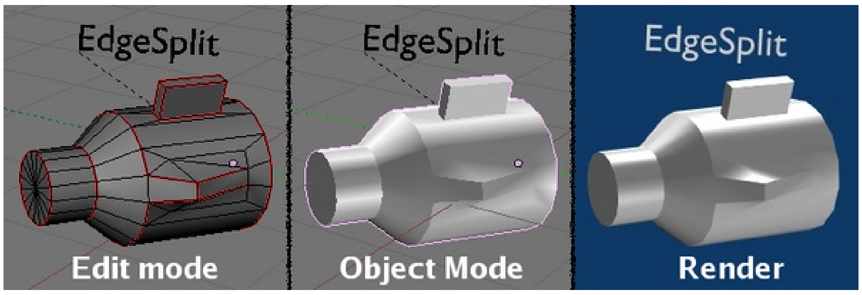
From Edge Angle
: If this button is enabled, edges will be split if their edge angle is greater than the Split Angle setting.

- The edge angle is the angle between the two faces which use that edge.
- If more than two faces use an edge, it is always split.
- If fewer than two faces use an edge, it is never split.

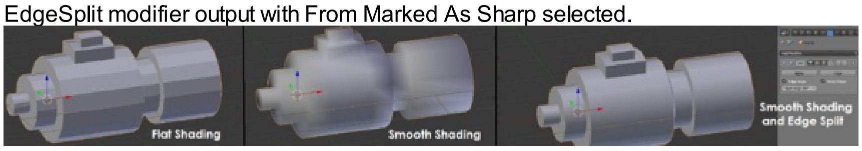
Split Angle
: This is the angle above which edges will be split if the From Edge Angle button is selected, from **0°** (all edges are split) to **180°** (no edges are split). This performs the same action as the Edge Specials » Edge Split menu item (CtrlE, in Edit mode)

From Marked As Sharp
: If this button is enabled, edges will be split if they are marked as sharp, using the Edge Specials » Mark Sharp menu item (CtrlE, in Edit mode).

Examples



EdgeSplit modifier output with From Marked As Sharp selected.



(From Left to right): Flat Shading, Smooth Shading, Smooth Shading with Edge Split.

Mask Modifier

Mode: Any mode

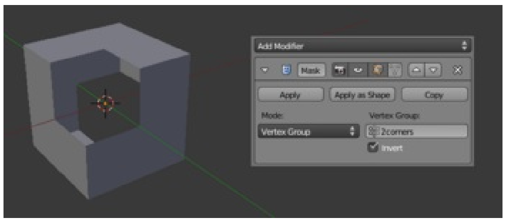
Panel: Modifiers

Description

The Mask modifier allows certain parts of an object's mesh to be hidden from view (masked off), in effect making the parts of the mesh that are masked act as if they were no longer there.

Options

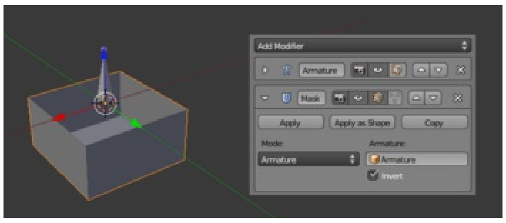
Mode
: The Mask modifier can hide parts of a mesh based on two different modes, selectable from this drop-down list:



Mode: Vertex Group
Vertex Group: 2corners
Invert

Vertex Group

Vertex Group
: When the Vertex Group option is selected, the Mask modifier uses the specified vertex group to determine which parts of the mesh are masked by the modifier.
Once you have entered the desired name, the Mask modifier will update so that those vertices of the mesh which are part of the named vertex group will be masked (which normally means they will be visible), and anything not part of the named vertex group will be made non-visible.
Any of the methods for assigning vertex weights to a mesh work with the Mask modifier; however the actual weight value assigned to a vertex group is completely ignored. The Mask modifier only takes into account whether a set of vertices are part of a group or not; the weight is not taken into account. So having a vertex group weight of say **0.5** will not make a partially masked mesh. Just being part of the vertex group is enough for the Mask modifier, even if the weight is **0.0**.



Mode: Armature
Armature: Armature
Invert

Armature

Armature
: Useful in Pose Mode or when editing an armature. Enter the name of the armature object in the text field. When working with bones in Pose mode, vertex groups not associated with the active bone are masked. The Inverse button can be useful to see how a bone affects the mesh down the chain of bones.

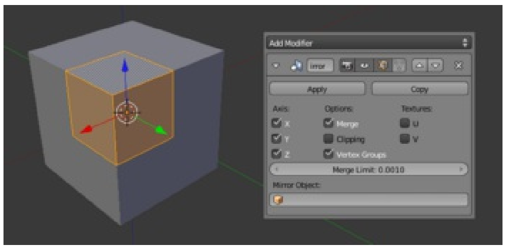
Inverse
: Normally, when the Mask modifier is applied to areas of a mesh, the parts that are under the influence of the modifier are left visible while the parts that aren’t are hidden. The Inverse button reverses this behavior, in that now parts of the mesh that were not originally visible become visible, and the parts that were visible become hidden.

Mirror Modifier

Mode: Any mode

Panel: Modifiers

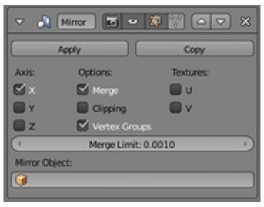
Description



The corner of a cube mirrored across three axes to form ... well ... a cube.

The Mirror modifier automatically mirrors a mesh along its **local** X, Y and/or Z axes, which pass through the object's center (the mirror plane is then defined by the two other axes). It can also use another object as mirror center, then use that object's local axes instead of its own. It can weld vertices together on the mirror plane within a specified *tolerance* distance. Vertices from the original object can be prevented from moving across or through the mirror plane. And last but not least, it can also mirror vertex groups and UV coordinates.

Options



Mirror modifier

Axis
: The axis (X, Y, or Z) along which to mirror (i.e. the axis perpendicular to the mirror plane of symmetry). To understand how the axis applies to the mirror direction, if you were to mirror on the X axis, the X plus values of the original mesh would become X minus values on the mirrored instance.
  You can select more than one of these axes – you'll then get more mirror instances, so that all planes of symmetry selected are "fully processed" (i.e. with one axis you get a single mirror, with two axes four mirrors, and with all three axes eight mirrors).

Options

Merge
: Merges vertices at the mirror plane. See Merge Limit below.

Clipping
: Prevents vertices from crossing through the mirror plane(s). Note that this is only valid in Edit mode (i.e when using object transformations, translations, scaling, et cetera, in Object mode, vertices will happily cross these borders.)
  If Clipping is selected but vertices are outside of the Merge Limit the vertices will not merge. As soon as the vertices are within Merge Limit they are clipped together and cannot be moved beyond the mirror plane. If several vertices are selected and are at different distances from the mirror plane, they will one by one be clipped at the mirror plane.
  Once you have confirmed clipped vertices with LMB you must, if you want to break the clipping, un-select Clipping to be able to move vertices away from the mirror.

Vertex Groups
: When this button is enabled, the Mirror modifier will try to mirror existing vertex groups. A nice feature that has otherwise specific prerequisites.

- First, the vertex groups you want to mirror must be named following the usual left/right pattern (i.e. suffixed by something like "`.R`", "`.right`", "`.L`", et cetera).
- Next, you must have the "mirrored" groups already existing (i.e. same names suffixed by the "other side") *and completely empty* (no vertex assigned to it), else it won't work.

  Usually, the mirrored copies of the vertices of a group remain in this group. Once this option is activated, all groups following the rules described above will only be valid on the original object – the mirrored copy will put these same vertices into the "mirror" group. Very handy with armatures, for example: you just model half of your object, carefully rig it with half of your armature, and just let the Mirror modifier build the other half. Just be sure to put your Armature modifier(s) after the Mirror one.
  A final word about multi-axes mirror: in these cases, the "direct", "first level" copies get the mirrored groups, the copies of copies ("second level") get the original groups, et cetera.

Textures
: The U and V options allows you to mirror, respectively, the U and V texture coordinates. The values are "mirrored" around the **0.5** value, i.e. if you have a vertex with UV coordinates of (**0.3**, **0.85**), its mirror copy will have UV coordinates of (**0.7**, **0.15**) with both buttons enabled.

Merge Limit
: The maximal distance between vertices and mirror plane for the welding between original and mirrored vertices to take place. The vertices then will snap together, allowing linking the original mesh to its mirrored copy.

Mirror Object
: The name of another object (usually an empty), to be used as the reference for the mirror process: its center and axes will drive the plane(s) of symmetry. You can of course animate its position/rotation (Ipo curves or others), to animate the mirror effect.

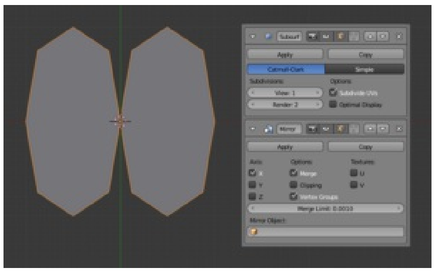
Hints

Many modeling tasks involve creating objects that are symmetrical. However, there used to be no quick way to model both halves of an object without using one of the workarounds that have been discovered by clever Blender artists over the years. A common technique is to model one half of an object and use AltD to create a linked duplicate which can then be mirrored on one axis to produce a perfect mirror-image copy, which updates in real time as you edit.

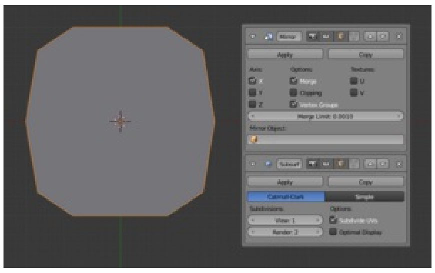
The Mirror modifier offers another, simpler way to do this. Once your modeling is completed you can either click Apply to make a real version of your mesh or leave it as is for future editing.

Using Mirror modifier with Subdivision Surface modifier

When using the Mirror modifier along with the Subsurf modifier, the order in which the modifiers are placed is important.


Subsurf modifier before Mirror modifier

This shows the Subsurf modifier placed before the Mirror one; as you can see the effect of this is that the mesh splits down the center line of the mirror effect.


Mirror modifier before Subsurf modifier

This shows the Mirror modifier placed before the Subsurf modifier. In this order you will get the the center line of the mesh snapped to the center line, which in most cases would be the desired effect.

Aligning for Mirror

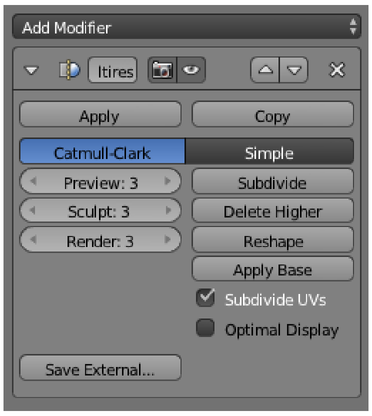
To apply a Mirror modifier, it is common to have to move the object's center onto the edge or face that is to be the axis for mirroring. This can be tricky when attempted visually. A good technique to achieve an exact position is to determine the edge against which you wish to mirror. Select two vertices on that edge. Then use ⇧ ShiftS followed by Cursor to Selection (C). This will center the 3D cursor exactly on the edge midway between the two vertices. Finally, press CtrlAlt⇧ ShiftC for the Set Origin popup, then select Origin to 3D Cursor (T). This will move the object's center to where the 3D cursor is located, and the mirroring will be exact.

An alternative is to use an Empty as a Mirror Object that you move to the correct position.

Multiresolution Modifier

Mode: Object mode

Panel: Properties Window -> Context Button Modifiers



Multires modifier

The Multiresolution modifier gives the ability to subdivide a mesh to different levels depending on whether you are viewing it from the 3D Viewport, Sculpt Mode or a Blender Render.

Options

Catmull-Clark / Simple
: Set the type of subdivision. Simple maintains the current shape, and simply subdivides edges. Catmull-Clark creates a smooth surface, smaller than the original.

Preview
: Set the level of subdivisions to use in the viewport.

Sculpt
: Set the number of subdivisions to use in Sculpt Mode.

Render
: Set the number of subdivisions to use when rendering.

Subdivide
: Add a higher level of subdivision.

Delete Higher
: Deletes all subdivision levels that are higher than the current one.

Reshape
: Copies vertex coordinates from another mesh. To use, select a different mesh with matching topology and vertex coordinates, then ⇧ Shift select the mesh and click Reshape. The mesh will take the shape of the other one.

Apply Base
: Modifies the mesh to match the form of the subdivided mesh.

Optimal Display
: Skips the drawing of edges added from subdivision.

Save External
: Saves displacements to an external .btx file.

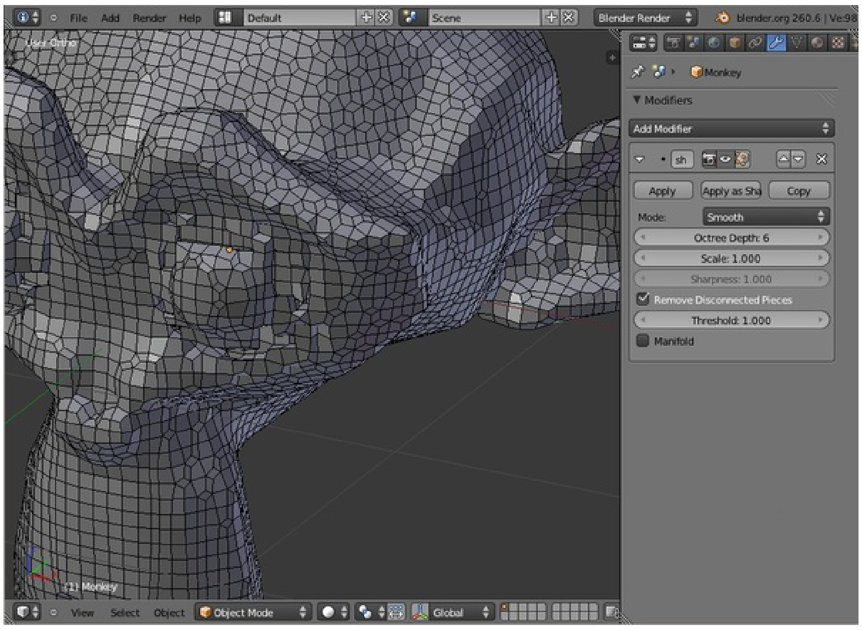
Remesh Modifier

Mode: Object mode

Panel: Properties Window -> Context Button Modifiers

Description

The Remesh modifier is a tool for generating new mesh topology based on an input surface. The output follows the surface curvature of the input, but its topology contains only quads.



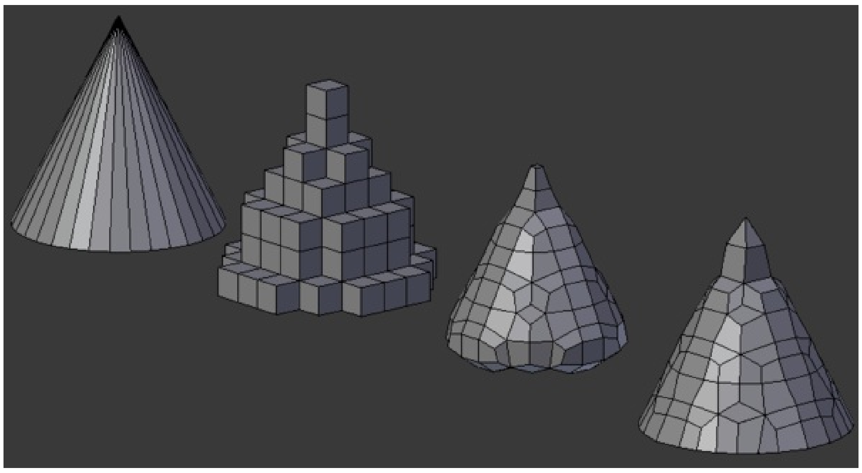
Mode: Smooth
Octree Depth: 6
Scale: 1.000
Sharpness: 1.000
Remove Disconnected Pieces [✓]
Threshold: 1.000
Manifold

Usage

In the modifier panel, add a Remesh modifier. (This modifier is only available for meshes.) The input mesh should have some thickness to it; if the input is completely flat, add a solify modifier above the remesh modifier.

Mode

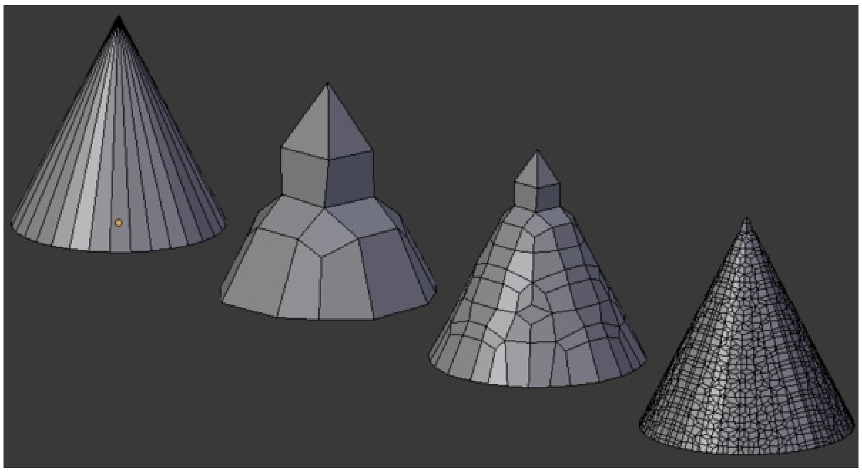
There are three basic modes available in the remesh modifier: Blocks, Smooth and Sharp.


This example shows a cone with each of the different remesh modes. From left to right: original cone, Blocks, Smooth, and Sharp

Note that the output topology is almost identical between the three modes; what changes is the smoothing. In Blocks mode, there is no smoothing at all. The Smooth and Sharp modes generate similar output in places where the mesh is already smooth, but Sharp mode preserves sharp details better. In this example, the circular bottom of the cone and the top point of the cone are accurately reproduced in Sharp mode.

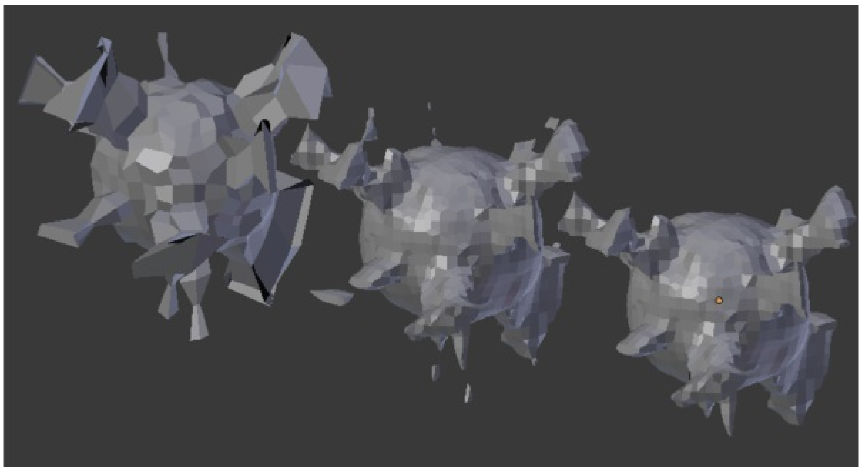
Octree Depth and Scale

The Octree Depth sets the resolution of the output. Low values will generate larger faces relative the input, higher values generate a denser output. The result can be tweaked further by setting the Scale; lower values effectively decrease the output resolution.


Input mesh, and the low to high resolution output meshes

Disconnected Pieces

To filter out small disconnected pieces of the output, enabled Remove Disconnected and set the threshold to control how small a disconnected component must be to be removed.


The input mesh (left) is fairly noisy, so the initial output of the remesh modifier (center) contains small disconnected pieces. Enabling Remove Disconnected Pieces (right) deletes those faces.

Sharpness

In Sharp mode, set the Sharpness value to control how closely the output follows sharp edges in the input (use lower values to filter out noise).

Demo Videos

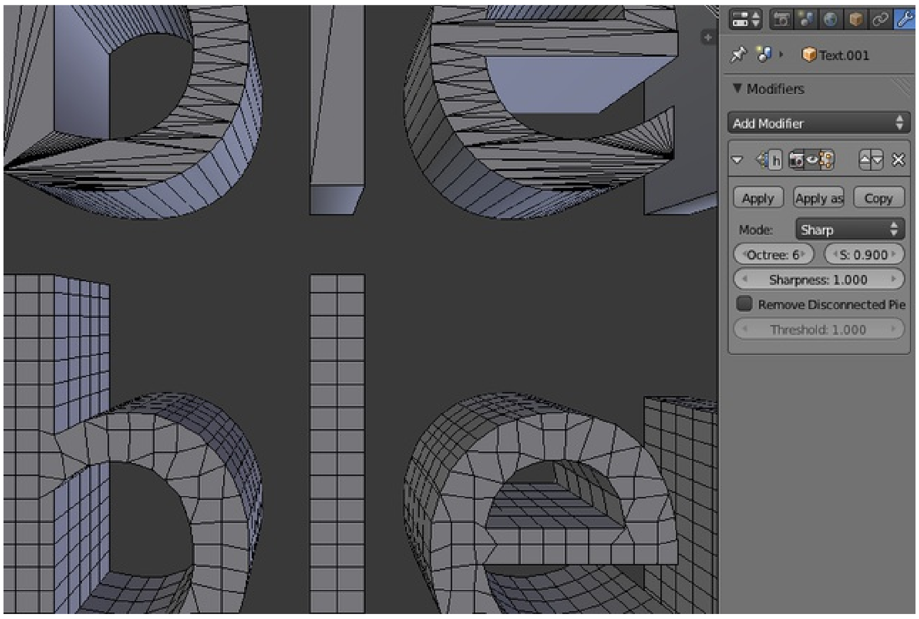
[video link]

[video link]

[video link]

[video link]

[video link] [video link]



Remesh modifier applied to text to improve topology

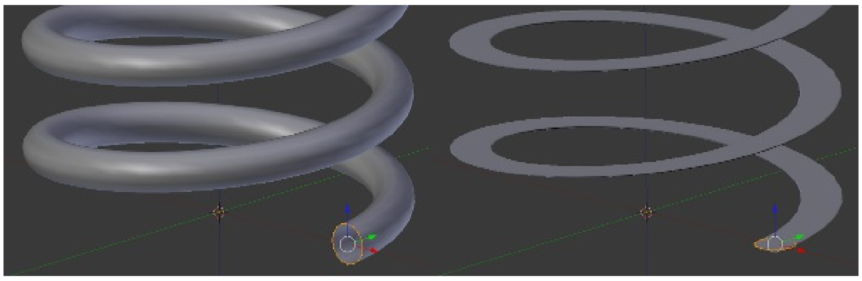
Screw Modifier

Mode: Any mode

Panel: Modifiers

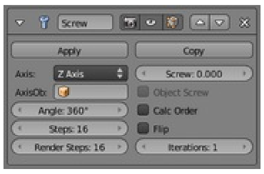
Description

The Screw modifier is similar to the Screw tool in the Tool Shelf in that it takes a profile object, a Mesh or a Curve, to create a helix-like shape.


Properly aligning the profile object is important

The profile should be two dimensional and properly aligned to the cardinal direction of the object rather than to the screw axis.

Options



Axis: Z Axis
Screw: 0.000
AxisOb:
Object Screw
Angle: 360°
Calc Order
Steps: 16
Flip
Render Steps: 16
Iterations: 1

Screw modifier

Axis
: The axis along which the helix will be built.

    Screw
    : The height of one helix iteration.

AxisOb
: The name of an object to define the axis direction.

    Object Screw
    : Use the Axis Object to define the value of Screw.

Angle
: Degrees for a single helix revolution.

Steps
: Number of steps used for a single revolution (displayed in the 3D view.)

Render Steps
: As above, used during render time. Increase to improve quality.

Calc Order
: Order of edges is calculated to avoid problems with normals. Only needed for meshes, not curves.

Flip
: Flip normals direction.

Iterations
: Number of revolutions.

Skin Modifier

Mode: Any mode

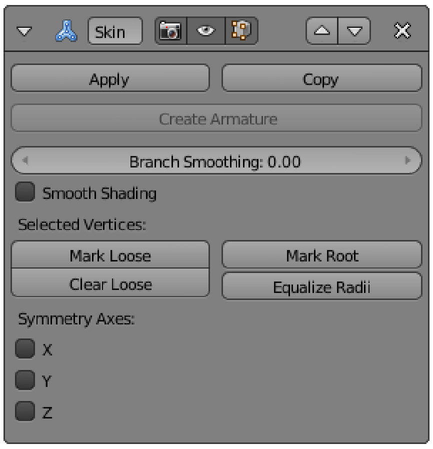
Panel: Modifiers

The Skin modifier uses vertices and edges to create a skinned surface, using a per-vertex radius to better define the shape.

It is a quick way to generate base meshes for sculpting and/or smooth organic shapes with arbitrary topology.

Note that faces in the input are ignored by the Skin modifier.

Options



Skin modifier UI.

Create Armature
: Create an armature for your object.

Branch Smoothing
: Smooth the intersection vertices.

Selected Vertices

Mark Loose
: Mark the selected vertices as loosing points. Loosing defines points which merge their faces with each other to fill gaps.

Clear Loose
: Take the Loose away. The initial behavior.

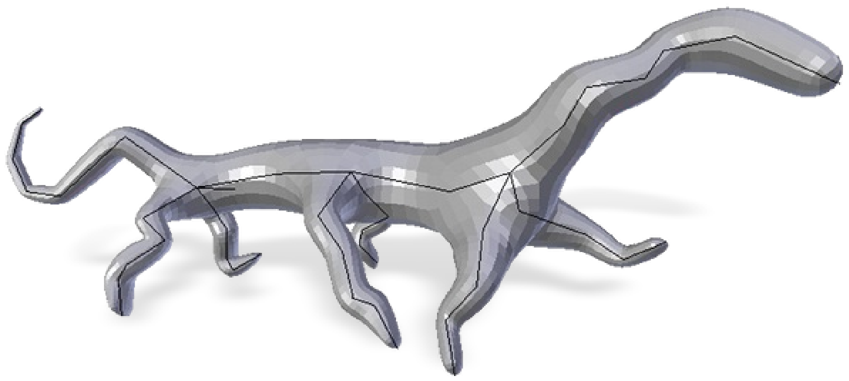
Mark Root
: Defines the root points of the object. These are the center of the object.

Equalize Radii
: Makes the skin radii of selected vertices equal on each axis.

Example

- Select the cube. ⇆ Tab into edit mode and AltM » At Center to merge all vertices at one point. E then Z to extrude the vertex along the Z axis.

Skin Node Set Flag
One of the mesh's vertices must be set to Root. If you by accident delete the default root vertex, select a vertex, hit the Skin Node Set Flag button, and in the Mesh Tools menu set the new vertex to root.



Simple creature, made with only the Skin modifier.

- In the modifiers' menu, add a Skin modifier.
- ⇆ Tab into edit mode and start extruding. To see the actual "Z spheres", Z to change to wireframe mode. These spheres are actual meshes with a lot of polygons, so performance issues might occur on older computers.
- Try to get sketch results similar to the picture (Simple creature, made with only the Skin modifier.), through extruding the vertices of the object.

- Use CtrlA to change the size of the different regions within the creature.
- Use Mark Loose at regions like the neck, to merge these faces more together.
- To get smoother results, activate Smooth Shading and use Ctrl3 on the object.

External links

- *Skin Modifier Development* at Blender Nation — An early demonstration of the skin modifier by Nicholas Bishop (March 2011)
- Ji, Zhongping; Liu, Ligang; Wang, Yigang (2010). *B-Mesh: A Fast Modeling System for Base Meshes of 3D Articulated Shapes*, Computer Graphics Forum 29(7), pp. 2169-2178. — The work this modifier is based on (direct link to PDF)
- Related thread on Blender artists

Solidify Modifier

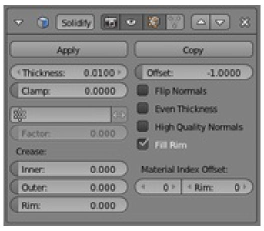
Mode: Any mode

Panel: Modifiers

Description

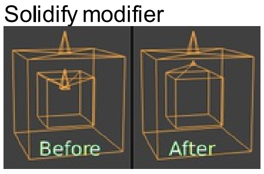
The Solidify modifier takes the surface of any mesh and adds a depth to it.

Options



Thickness: 0.0100
Clamp: 0.0000
Factor: 0.000
Crease:
Inner: 0.000
Outer: 0.000
Rim: 0.000
Offset: -1.0000
Flip Normals
Even Thickness
High Quality Normals
✓ Fill Rim
Material Index Offset:
0 Rim: 0

Solidify modifier



Clamp Offset

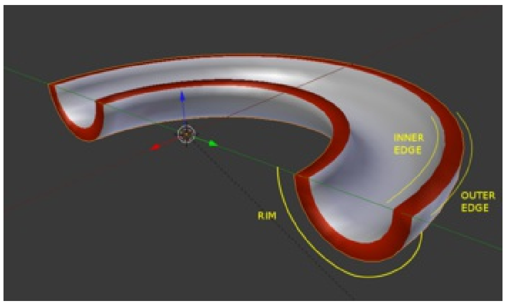
Thickness
: The depth to be solidified.

Offset
: A value between **-1** and **1** to locate the solidified output inside or outside the original mesh. Set to zero, Offset will center the solidified output on the original mesh.

Clamp
: A value between **0** and **2** to clamp offsets to avoid self intersection.

Vertex Group
: Restrict the modifier to only this vertex group.

    Invert
    : Inverts the previous selection.



Rim and edges. In this example, the object was assigned a second material used to color the rim red.

Crease
: These options are intended for usage with the Subdivision Surface modifier.

    Inner
    : Assign a crease to the inner edges.

    Outer
    : Assign a crease to the outer edges.

    Rim
    : Assign a crease to the rim.

Even Thickness
: Maintain thickness by adjusting for sharp corners. Sometimes improves quality but also increases computation time.

High Quality Normals
: Normals are calculated to produce a more even thickness. Sometimes improves quality but also increases computation time.

Fill Rim
: Fills the gap between the inner and outer edges.

Rim Material
: Uses the object's second material for the rim; this is applied as an offset from the current material.

Hints

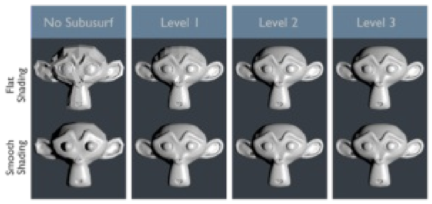
- The modifier thickness is applied before object scale; if maintaining a fixed thickness is important use unscaled objects (or account for the scale).
- Solidify thickness is an approximation. While "Even Thickness" and "High Quality Normals", should yield good results, the architectural/CAD modeling the final wall thickness isn't guaranteed, depending on the mesh topology. To look at it differently - maintaining precise wall thickness in some cases would need to add / remove faces on the offset shell - something this modifier doesn't do since this would add a lot of complexity and slow down the modifier.

Subdivision Surfaces ("Subsurf") Modifier

Mode: Any Mode

Panel: Modifiers > Add > Generate > Subdivision Surface

Hotkey: Using the regular #s, not keypad #s: Ctrl0, Ctrl1, Ctrl2, Ctrl3, Ctrl4, Ctrl5. Note, this only affects the View value while leaving Render at 2 (see below).



Click to zoom; Subsurfs levels 0 to 3, without and with Smooth Shading. This was rendered from: SubsurfDemo.blend

Subdivision Surface (Subsurf in short) is a method of subdividing the faces of a mesh to give a smooth appearance, to enable modeling of complex smooth surfaces with simple, low-vertex meshes. This allows high resolution mesh modeling without the need to save and maintain huge amounts of data and gives a smooth *organic* look to the object.

This process creates virtual geometry that is generated at display time, but it can be converted to real geometry that you could edit with the Apply feature. Like the rest of the modifiers, this is a non-destructive tool, not affecting the underlying mesh until you hit Apply (and even then, you have Undo/Redo capabilities).

Also, like the rest of the Modifiers, order of execution has an important bearing on the results. For this, see the Order of the Modifier Stack section on this page.

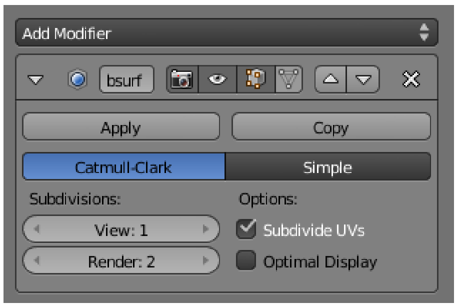
There are two algorithms available: Simple (subdivides mesh) and the default Catmull-Clark (subdivides and smooths mesh).

Keep in mind that this is a different operation than its companion, Smooth Shading. You can see the difference between the two in the grid image to the right.

MultiResolution Modifier

Another way to subdivide is with the MultiResolution Modifier. This differs from Subsurf in that MultiRes allows you to edit the mesh at several subdivision levels without losing information at the other levels. It is slightly more complicated to use, but more powerful.

Options



Modifier's panel

Subsurf is a modifier. To add it to a mesh, press Add Modifier and select Subdivision Surface from the list.

Type
: This toggle button allows you to choose the subdivision algorithm:

  Catmull-Clark
  : The default option, subdivides and smooths the surfaces. According to its Wikipedia page, the "arbitrary-looking formula was chosen by Catmull and Clark based on the aesthetic appearance of the resulting surfaces rather than on a mathematical derivation."

  Simple
  : Only subdivides the surfaces, without any smoothing (similar to W → Subdivide, in Edit mode). Can be used, for example, to increase base mesh resolution when using displacement maps or textured emitters with indirect lighting.

Subdivisions
: Recursively adds more geometry. For some detailed examples of the numbers, see the Performance Considerations section.

  View
  : Affects the display resolution for the 3D views only.

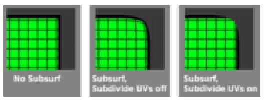
Render

Affects the subdivision level used during rendering. For the internal Blender Render, the status line at the top of the Render Result will tell you the current Frame, then after that the number of the final, generated vertices and faces. This can give you a clue at the overall performance impact of all Modifiers.

The right combination of these settings will allow you to keep a fast and lightweight approximation of your model when interacting with it in 3D, but use a higher quality version when rendering.

View less than or equal to Render

Be careful not to set the View higher than the Render setting, or else your preview would display higher quality than your render.



Subdivide UVs on and off -- see the .blend for the source of this image.

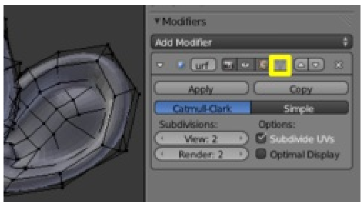
Options

Subdivide UVs

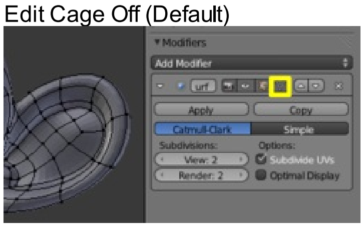
When enabled, the UV maps will also be subsurfed (i.e. Blender will add “virtual” coordinates for all sub-faces created by this modifier). The easiest way to understand its effects is to view Manual-Modifiers-Generate-Subsurf-SubdivideUVsExample.blend.

Optimal Display

Restricts the wireframe display to only show a warped mesh cage edges, rather than the subdivided result, to help visualization. Without this, Edit Mode can look cluttered with lines that are not really there.



Edit Cage Off (Default)



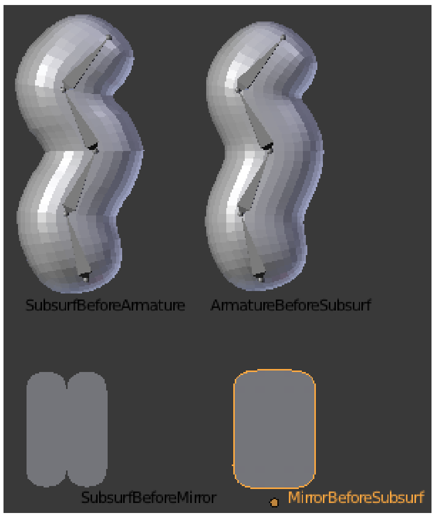
Edit Cage On

Edit Cage Mode

To view and edit the results of the subdivision (“isolines”) while you’re editing the mesh, you must enable the Editing Cage mode by clicking in the inverted triangle button in the modifier panel header (next to the arrows for moving the modifier up and down the stack). This lets you grab the points as they lie in their new subdivided locations, rather than on the original mesh.

Notice the comparison of screenshots to the right. With the edit cage off, some vertices are buried under the subsurfed mesh. With dense vertex configurations, you might have to even click the "Eye" icon so that you can see these vertices. The "edit cage on" version does not suffer from this problem. It does, however, have its own disadvantage---it can look *too* nice. Notice the three quads running in the middle of Suzanne's ear. You can only tell how crooked they are in the "edit cage off" version. When you are modelling, you will more often want to see your mesh deformities in their full ugliness---so you can apply your skills until it is sheer prettiness.

Order of the Modifier Stack



Notice that the Armature Modifier before the Subsurf comes out much better in this case. Also, the Mirror before the Subsurf is clearly correct compared to the other way around.

The Evaluation order of Modifiers is often significant, but especially so in the case of the Subsurf. The key to deciding your Modifier stack order is to picture the changes at each step, perhaps by temporarily Apply'ing the Modifiers, or perhaps by simply tinkering with the order until things come out right. To see the file behind these screenshots, you can look at Manual-Modifiers-Generate-Subsurf_OrderOfExecution.blend.

Control

Subsurf rounds off edges, and often this is not what you want. There are several solutions.

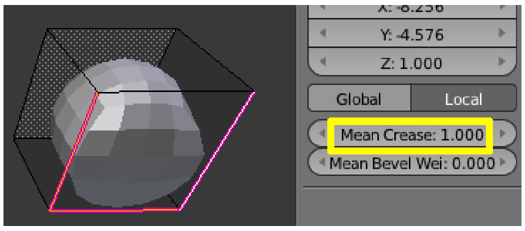
Weighted Creases

Mode: Edit Mode (Mesh)

Panel: 3D View → Transform Properties

Hotkey: ⇧ ShiftE

Menu: Mesh → Edges → Crease Subsurf



A Subsurfed Cube with Creased Edges

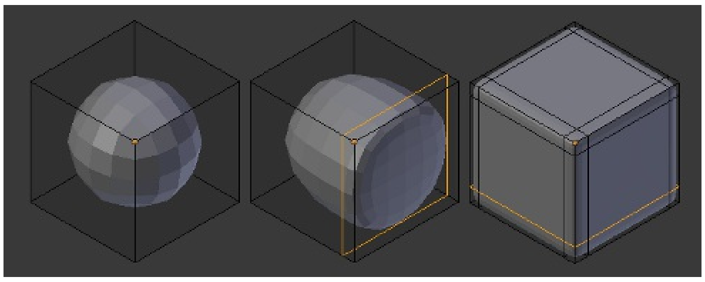
Weighted edge creases for subdivision surfaces allows you to change the way Subsurf subdivides the geometry to give the edges a smooth or sharp appearance.

The crease weight of selected edges can be changed using Transform Properties (N) and change the Median Transform slider. A higher value makes the edge “stronger” and more resistant to subsurf. Another way to remember it is that the weight refers to the edge’s sharpness. Edges with a higher weight will be deformed less by subsurf. Recall that the subsurfed shape is a product of all intersecting edges, so to make the edges of an area sharper, you have to increase the weight of all the surrounding edges.

Edge Loops

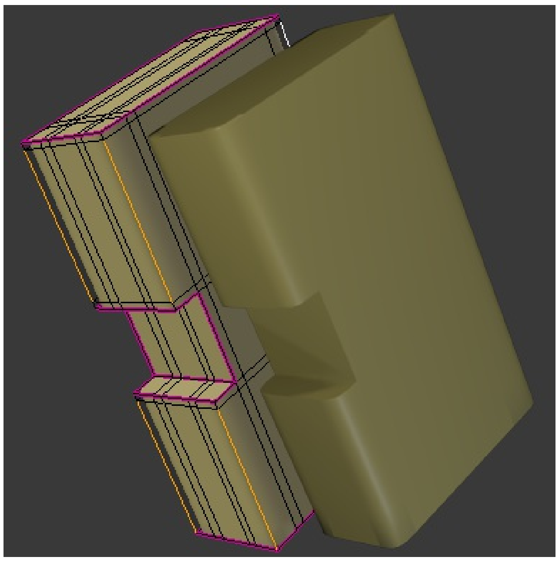
Mode: Edit Mode (Mesh)

Hotkey: CtrlR


A Subsurf Level 2 Cube, the same with an Edge Loop, and the same with six Edge Loops

The Subsurf modifier demonstrates why good, clean topology is so important. As you can see in the figure, the Subsurf modifier has a drastic effect on a default Cube. Until you add in additional Loops (with CtrlR), the shape is almost unrecognizable. A mesh with a deliberate topology has good placement of Edge Loops, which allow the placement of more Loops (or removal of Loops, with X » Edge Loop) to control the sharpness/smoothness of the resultant mesh.

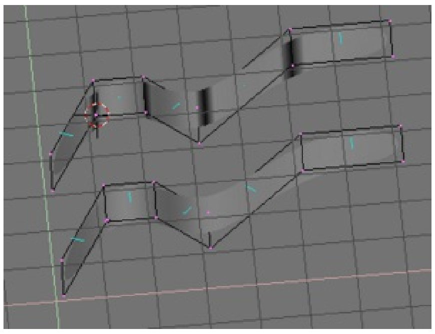
Combination

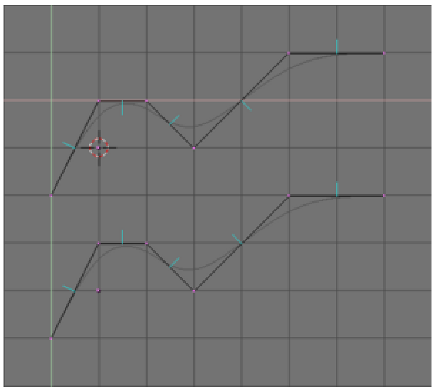

Purple edges are Creased, Orange are intended to be rounded off. See: File:WoodBlock.blend

It is valuable to know the use of all three tools: Smooth/Flat Shading, Edge Creases and Edge Loops. Consider the task of modelling a 2"x4" block of wood that has had a notch cut out. The factory edges are rounded off (a good task for Smooth Shading and some Edge Loops), but the edges where the saw touched are crisp (a good task for Flat Shading and Edge Crease). Note that we had to add some extra edge loops near the Creased edges -- this was only to limit the effects of the Smooth Shading, which bleeds over onto the flat faces.

Limitations & Workarounds

Blender's subdivision system produces nice smooth subsurfed meshes, but any subsurfed face (that is, any small face created by the algorithm from a single face of the original mesh), shares the overall normal orientation of that original face.

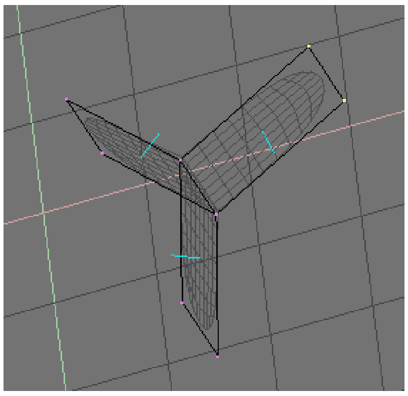

Solid view of subsurfed meshes with inconsistent normals (top) and consistent normals (bottom). Note the ugly dark areas that appear.


Side view of the above meshes' normals, with random normals (top) and with coherent normals (bottom).

Abrupt normal changes can produce ugly black gouges (See: *Solid viewof subsurfed meshes with inconsistent normals (top) and consistent normals (bottom)*), even though these flipped normals are not an issue for the shape itself (See: *Side viewof subsurfed meshes*).

A quick way to fix this (one which works 90% of the time) is to use Blender's "Make Normals Consistent" operation: In Edit Mode, select all with A, then hit CtrlN to recalculate the normals to point outside. If you still have some ugly black gouges after a CtrlN, you will have to manually flip some normals. To do this (still in Edit Mode), look in the N Properties Panel, on the right, in the Mesh Display subsection (it is roughly the 3rd up from the bottom). There you can turn on the little cyan "Face Normals" sticks (as seen in our pictures to the right), and you can change their size to be more appropriate for the scale of your mesh. If you then go around your mesh in Face Select mode (Ctrl⇆ Tab,F) selecting bad faces, you can then use the Specials » Flip Normals functionality (shortcut: W,N) to fix them. Smoothing out normals is good for the mesh, and it's good for the soul.

Note that one problem that will prevent Blender from automatically recalculate normals correctly is if the mesh is not “Manifold”. A “Non-Manifold” mesh contains an edge that is not connected to (exactly) two faces. Generally, this means that “out” cannot be computed.


A “Non-Manifold” mesh.

(*A “Non-Manifold” mesh*) shows a very simple example of a “Non-Manifold” mesh. In general a non-manifold mesh occurs when you have internal faces or edges that are unexpectedly open.

A non-manifold mesh is not a problem for conventional meshes, but can give rise to ugly artifacts when subsurfed. Also, it does not allow decimation, so it is better to avoid them as much as possible.

To locate the non-Manifold components, you can be in either Vertex Select mode or Edge Select mode and deselect all vertices. Now, either go to Select » Non Manifold or hit CtrlAlt⇧ ShiftM. Sometimes, it can take some clever work to make these areas Manifold, but with determination and creativity you will be able to figure it out. Sometimes it is only a matter of Removing Doubles (W,R) or of manually merging some vertices (AltM).

Performance Considerations

Great levels of Subsurf demands more video memory, and a faster graphics card. Blender could potentially crash if your level of Subsurf surpasses your system memory.

Note about potential crashes: Be aware that the Subsurf Modifier will need more and more memory at higher levels of subsurf, and the more dense your base mesh, the more memory you will need. In 32 bit systems, Blender could potentially crash with *malloc* errors, when you surpass 2~3 GiB of memory used. This is not a Blender bug. Blender, when paired with a 64 bit system, could use 64 GiB of memory, thus reducing the chances of *malloc()* errors.

Another note about using high levels of Subsurf with a graphics card with a low total amount of Vram: When you move, edit, or otherwise work in your mesh, some parts of the geometry will disappear visually. Your mesh will actually be O.K., because the render is generated using your Object Data, even though it cannot be shown by your graphics card.

The resulting Vertex, Edge, and Face counts from the Modifier's effect on a Cube, Quadrilateral Plane, and Triangle can be found in these tables:

Cube Subdivision Level	Resulting Verts	Resulting Edges	Resulting Faces
0	8	12	6
1	26	48	24
2	98	192	96
3	386	768	384
4	1538	3072	1536
5	6146	12288	6144
6	24578	49152	24576
Formulae	3*2**(2*n)+4)/2	3*4**n	verts - 2

While we're at it, here is the pattern for subdividing a quadrilateral plane:

Quad Subdivision Level	Resulting Verts	Resulting Edges	Resulting Faces
0	4	4	1
1	9	12	4
2	25	40	16
3	81	144	64
4	289	544	256
5	1089	2112	1024
6	4225	8320	4096
Formulae	((2**n+2)**2)/4	2**(n-1)*(2**n+2)	4**(n-1)

And, of course, triangles:

Tri Subdivision Level	Resulting Verts	Resulting Edges	Resulting Faces
0	3	3	1
1	7	9	3
2	19	30	12
3	61	108	48
4	217	408	192
5	817	1584	768
6	3169	6240	3072
Formulae	Do you know it?	3*(2**(n-3))*(2**n+2)	3*4**(n-1) [Rounding n=0 case up to int]

Category:Modifiers]]

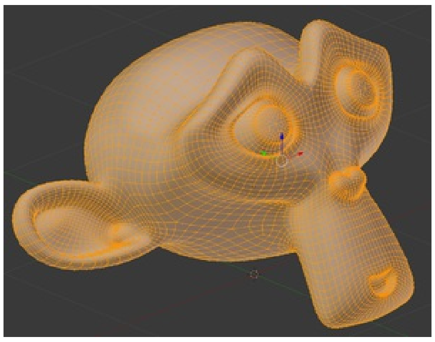
Triangulate Modifier

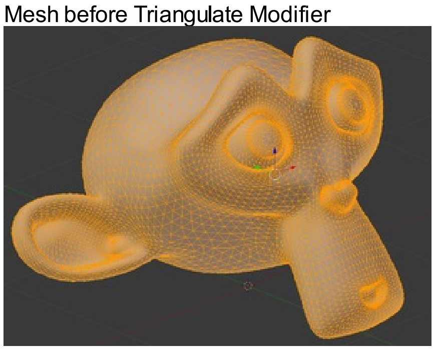
Mode: Object mode

Panel: Properties Window -> Context Button Modifiers

Description

The Triangulate modifier converts all faces in a mesh (whether it be quads or N-gons) to triangular faces. This modifier does the exact same function as the Ctrl+T triangulate in Edit Mode


Mesh before Triangulate Modifier


Mesh after Triangulate Modifier

Options

Quad Method

- Beauty
  - Split the quads in nice triangles, slower method.
- Fixed
  - Split the quads on the 1st and 3rd vertices.
- Fixed Alternate
  - Split the quads on the 2nd and 4th vertices.
- Shortest Diagonal
  - Split the quads based on the distance between the vertices.

Ngon Method

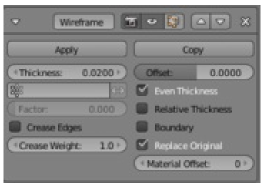
- Beauty
  - Arrange the new triangles nicely, slower method.
- Scanfill
  - Split the ngons using a scanfill algorithm.

Wireframe Modifier

Mode: Any mode

Panel: Modifiers

Description

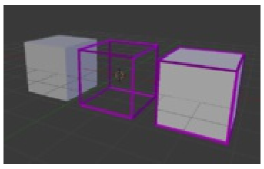


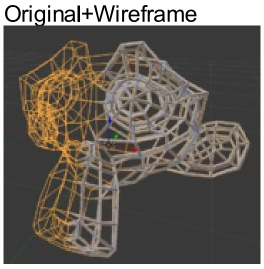
Thickness: 0.0200
Offset: 0.0000
[✓] Even Thickness
Factor: 0.000
[ ] Relative Thickness
[ ] Crease Edges
[ ] Boundary
Crease Weight: 1.0
[✓] Replace Original
Material Offset: 0

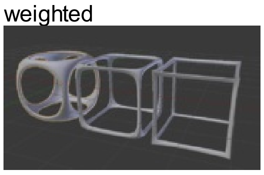
Wireframe Modifier

The Wireframe modifier transforms a mesh into a wireframe by iterating over its faces, collecting all edges and turning those edges into 4 sided polygons. Be aware of the fact that your mesh needs to have faces to be wireframed. You can define the thickness, the material and several other parameters of the generated wireframe dynamically via the given modifier options.

When you got more Faces that meet at one point they are forming a star like pattern like seen in the examples below.


Original / Wireframe / Original+Wireframe


VGroup weighting: One half 0 weighted, one half 1 weighted


Cube+Subsuf with 0 / 0.5 / 1 crease weight

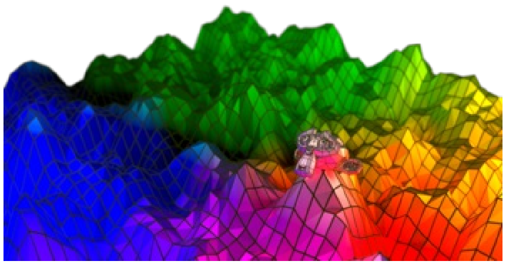
Options

Thickness
: The depth or size of the wireframes.

Offset
: A value between **-1** and **1** to change whether the wireframes are generated inside or outside the original mesh. Set to zero, Offset will center the wireframes around the original edges.

Vertex Group
: Restrict the modifier to only this vertex group.

    Invert
    : Inverts the vertex group weights.


Wireframes on a displaced plane. In this example, the wireframes carry a second (dark) material while the displaced plane uses its original one.

Crease Edges
: This option is intended for usage with the Subdivision Surface modifier. Enable this option to crease edges on their junctions and prevent large curved intersections.

  Crease Weight
  : Define how much crease (between **0** = no and **1** = full) the junctions should receive.

Even Thickness
: Maintain thickness by adjusting for sharp corners. Sometimes improves quality but also increases computation time.

Relative Thickness
: Determine edge thickness by the length of the edge - longer edges are thicker.

Boundary
: Creates wireframes on mesh island boundaries.

Replace Original
: If this option is enabled, the original mesh is replaced by the generated wireframe. If not, the wireframe is generated on top of it.

Material Offset
: Uses the chosen material index as as the material for the wireframe; this is applied as an offset from the first material.

Hints

- Wireframe thickness is an approximation. While **Even Thickness** should yield good results in many cases, skinny faces can cause ugly spikes, in this case you can either reduce the extreme angles in the geometry or disable the **Even Thickness** option.

Armature Modifier

Mode: Object mode

Panel: Modifiers

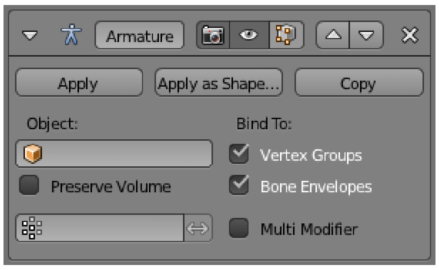
Description

The Armature modifier is used for building skeletal systems for animating the poses of characters and anything else which needs to be posed. With the Vertex Group and Multi Modifier options, you can use several armatures to animate a single (mesh) object.

By adding an armature system to an object, that object can be deformed accurately so that geometry doesn’t have to be animated by hand. The Armature modifier allows objects to be deformed by bones simply by specifying the name of the armature object, without having to use the (old) “parent/child” system.

For more details on armatures usage, see this chapter.

Options



Armature modifier

Object
: The name of the armature object used by this modifier.

    Preserve Volume
    : Use quaternions for preserving volume of object during deformation. It can be better in many situations.

    Vertex Group
    : The name of a vertex group of the object, the weights of which will be used to determine the influence of this Armature modifier’s result when mixing it with the results from other Armature ones. Only meaningful when having at least two of these modifiers on the same object, with Multi Modifier activated.

    Multi Modifier
    : Use the same data as a previous (Armature?) modifier as input. This allows you to use several armatures to deform the same object, all based on the “non-deformed” data (i.e. this avoid having the second Armature modifier deform the result of the first one…). The results of the Armature modifiers are then mixed together, using the weights of the VGroup vertex groups as “mixing guides”.

Bind To
: Method to bind the armature to the mesh.

    Vertex Groups
    : Enable/Disable vertex groups defining the deformation (i.e. bones of a given name only deform vertices belonging to groups of same name).

    Bone Envelopes
    : Enable/Disable bone envelopes defining the deformation (i.e. bones deform vertices in their neighborhood).

    Invert
    : Inverts the influence set by the vertex group defined in previous setting (i.e. reverts the weight values of this group).

Cast Modifier

Mode: Any mode

Panel: Modifiers

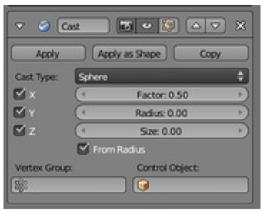
This modifier shifts the shape of a mesh, curve, surface or lattice to any of a few pre-defined shapes (sphere, cylinder, cuboid).

It is equivalent to the To Sphere tool in the Editing context (Mesh ? Transform ? To Sphere Alt⇧ ShiftS) and what other programs call “Spherify” or “Spherize”, but, as written above, it is not limited to casting to a sphere.

Hint
The Smooth modifier is a good companion to Cast, since the cast shape sometimes needs smoothing to look nicer or even to fix shading artifacts.

Important
For performance, this modifier works only with local coordinates. If the modified object looks wrong, you may need to apply the object’s rotation (CtrlA), especially when casting to a cylinder.

Options



Cast Type: Sphere
X ✓
Y ✓
Z ✓
Factor: 0.50
Radius: 0.00
Size: 0.00
✓ From Radius

cast modifier

Cast Type
: Menu to choose cast type (target shape): Sphere, Cylinder or Cuboid.

X, Y, Z
: Toggle buttons to enable/disable the modifier in the X, Y, Z axes directions (X and Y only for Cylinder cast type).

Factor
: The factor to control blending between original and cast vertex positions. It’s a linear interpolation: **0.0** gives original coordinates (i.e. modifier has no effect), **1.0** casts to the target shape. Values below or above `[0.0, 1.0]` deform the mesh, sometimes in interesting ways.

Radius
: If non-zero, this radius defines a sphere of influence. Vertices outside it are not affected by the modifier.

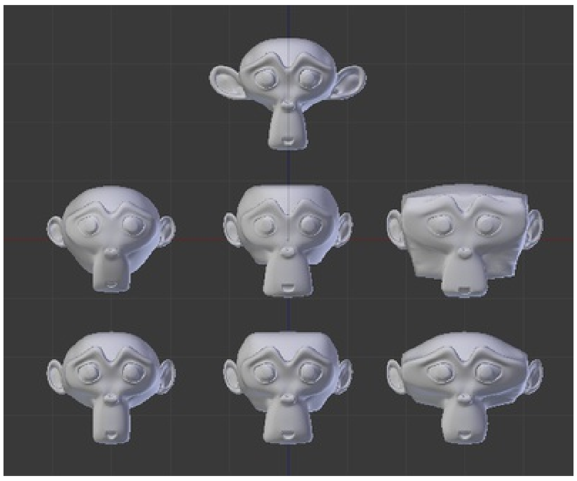
Size
: Alternative size for the projected shape. If zero, it is defined by the initial shape and the control object, if any.

From radius
: If activated, calculate Size from Radius, for smoother results.

Vertex Group
: A vertex group name, to restrict the effect to the vertices in it only. This allows for selective, real-time casting, by painting vertex weights.

Control Object
: The name of an object to control the effect. The location of this object’s center defines the center of the projection. Also, its size and rotation transform the projected vertices. Hint: animating (keyframing) this control object also animates the modified object.

Example


Top: Suzanne without modifiers. Middle: Suzanne with each type of Cast Modifier (Sphere, Cylinder and Cuboid). Bottom: Same as above, but now only X axis is enabled. Sample blend file

Curve Modifier

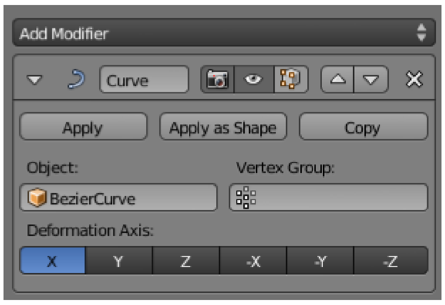
Mode: Object mode

Panel: Modifiers

The Curve Modifier provides a simple but efficient method of defining a deformation on a mesh based on the position of an curve object.

The Curve Modifier works on a (global) dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction, the mesh will traverse along the curve. Moving the mesh in an orthogonal direction will move the mesh object closer or further away from the curve. The default settings in Blender map the Y axis to the dominant axis. When you move the object beyond the curve endings the object will continue to deform based on the direction vector of the curve endings.

Options



Curve modifier

Object
: The name of the curve object that will affect the deformed object.

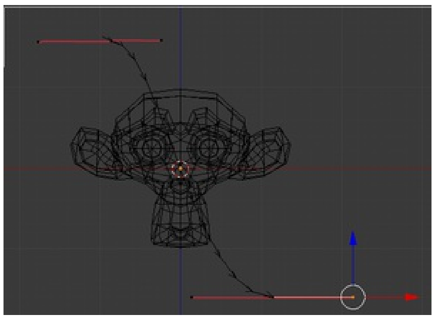
Vertex Group
: A vertex group name within the deformed object. The modifier will only affect vertices assigned to this group.

Deformation Axis
: X, Y, Z, -X, -Y, -Z: This is the axis that the curve deforms along.

Example

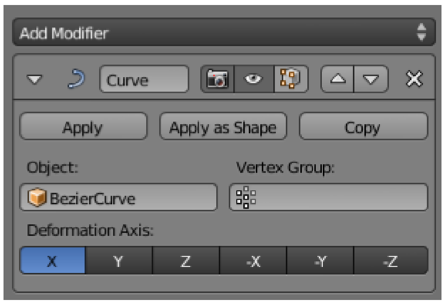
Let’s make a simple example:

- Remove default cube object from scene and add a Monkey (⇧ ShiftA → Add → Mesh → Monkey, *Add a Monkey!*)!
- Now add a curve (⇧ ShiftA → Add → Curve → Bezier Curve, *Add a Curve*).

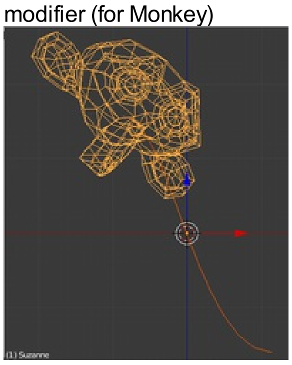

Edit Curve.

- While in Edit mode, move the control points of the curve as shown in (*Edit Curve*), then exit Edit mode (⇆ Tab).

- Select the Monkey (RMB) in Object mode
- Assign the curve to the modifier, as shown below. The Monkey should be positioned on the curve:



Assign the Bezier curve to the Curve modifier (for Monkey)



Monkey on a Curve.

- Now if you select the Monkey (RMB), and move it (G), in the Y-direction, the monkey will deform nicely along the curve.

A Tip

If you press MMB while moving the Monkey you will constrain the movement to one axis only.

Monkey deformations.

- In the image to the right you can see the Monkey at different positions along the curve. To get a cleaner view over the deformation SubSurf got applied with Subdiv to **2**, and Set Smooth on the Monkey mesh.

Displace Modifier

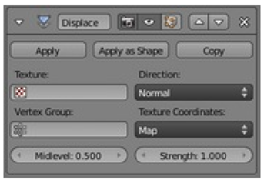
Mode: Any mode

Panel: Modifiers

Description

The Displace modifier displaces vertices in a mesh based on the intensity of a texture. Either procedural or image textures can be used. The displacement can be along a particular local axis, along the vertex normal, or the separate RGB components of the texture can be used to displace vertices in the local X, Y and Z directions simultaneously.

Options



Texture:
Direction: Normal
Vertex Group:
Texture Coordinates: Map
Midlevel: 0.500
Strength: 1.000

Displace modifier

Texture
: The name of the texture from which the displacement for each vertex is derived.
If this field is empty, the modifier defaults to 1.0 (white).

Vertex Group
: The name of a vertex group which is used to control the influence of the modifier.
If VGroup is empty, the modifier affects all vertices equally.

Midlevel
: The texture value which will be treated as no displacement by the modifier. Texture values below this value will result in negative displacement along the selected direction, while texture values above this value will result in positive displacement. This is achieved by the equation `(displacement) = (texture value) - Midlevel`.
Recall that color/luminosity values are typically between **0.0** and **1.0** in Blender, and not between **0** and **255**.

Direction
: The direction along which to displace the vertices.
Can be one of the following:

- X – displace along local X axis.
- Y – displace along local Y axis.
- Z – displace along local Z axis.
- RGB -> XYZ – displace along local XYZ axes individually using the RGB components of the texture.
- Normal – displace along vertex normal.

Texture Coordinates
: The texture coordinate system to use when retrieving values from the texture for each vertex.
Can be one of the following:

- UV – take texture coordinates from face UV coordinates.

  UV Layer
  : The UV coordinate layer from which to take texture coordinates.
  If the object has no UV coordinates, it uses the Local coordinate system. If this field is blank, but there is an UV coordinate layer available (e.g. just after adding the first UV layer to the mesh), it will be overwritten with the currently active UV layer.

Note
Since UV coordinates are specified per face, the UV texture coordinate system currently determines the UV coordinate for each vertex from the first face encountered which uses that vertex; any other faces using that vertex are ignored. This may lead to artifacts if the mesh has non-contiguous UV coordinates.

- Object – take the texture coordinates from another object’s coordinate system (specified by the Object field).

  Object
  : The object from which to take texture coordinates. Moving the object will therefore alter the coordinates of the texture mapping. Take note that moving the original object will **also** result in a texture coordinate update. As such, if you need to maintain a displacement coordinate system while moving the object to which the displacement is set, you will also have to move the related object at the same rate and direction.
  If this field is blank, the Local coordinate system is used.

- Global – take the texture coordinates from the global coordinate system.

- Local – take the texture coordinates from the object’s local coordinate system.

Strength
: The strength of the displacement. After offsetting by the Midlevel value, the displacement will be multiplied by the Strength value

to give the final vertex offset. This is achieved by the equation `(vertex_offset) = (displacement) × Strength`.
A negative strength can be used to invert the effect of the modifier.

See also

- Blender artists post: Displace modifier tutorial (September 2006)

Hook Modifier

Mode: Any mode

Panel: Modifiers

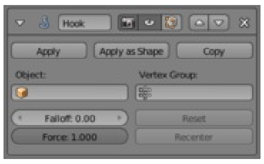
Description

Two spheres used as Hooks to deform a subdivided cube.

The Hook modifier is used together with Shape Keys to control the deformation of a Mesh or a Lattice using another object (usually an Empty but it can be any object).

As the hook moves, it weighted-pulls vertices from the mesh with it. If you have used proportional editing, you can think of it as animated proportional editing. While hooks do not give you the fine control over vertices movement that shape keys do, they are much simpler to use.

Options



Hook modifier

Object
: The parent object name for the hook.

Falloff
: If not zero, the falloff is the distance where the influence of a hook goes to zero. It uses a smooth interpolation, comparable to the proportional editing tool.

Force
: Since multiple hooks can work on the same vertices, you can weight the influence of a hook this way. Weighting rules are:

  - If the total of all forces is smaller than **1.0**, the remainder (`1.0 - (forces sum)`, will be the factor the original position has as its force.
  - If the total of all forces is larger than **1.0**, it only uses the hook transformations, averaged by their weights.

The following settings are only available in Edit mode:

Reset
: Recalculate and clear the offset transform of hook.

Recenter
: Set hook center to cursor position.

Select
: Select affected vertices on mesh.

Reassign
: Reassigns selected vertices to this hook.

Hints

- The hook modifier stores vertex indices from the original mesh to determine what to effect; this means that modifiers that generate geometry, like subsurf, should always be applied **after** the hook modifier; otherwise the generated geometry will be left out of the hook's influence.

Laplacian Smooth and Shape Enhanced Modifier

Mode: Any mode

Panel: Modifiers

Description

The Laplacian Smooth and Shape Enhanced modifier allows you to reduce noise on a mesh's surface with minimal changes to its shape, and exaggerates a shape using a Laplacian smoothing modifier in the reverse direction using a single parameter, ***factor***, that supports negative and positive values: **negative for enhancement** and **positive for smoothing**.

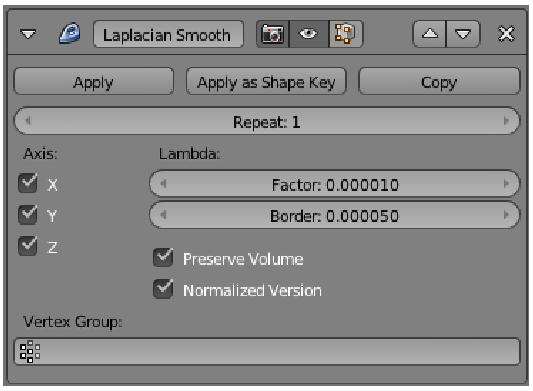
The Shape enhanced method exaggerates a shape using a Laplacian smoothing operator in the reverse direction.

The Laplacian Smooth is useful for objects that have been reconstructed from the real world containing undesirable noise. A mesh smoothing tool removes noise while still preserving desirable geometry as well as the shape of the original model.

The Laplacian Smooth and Shape Enhanced modifier is based on a curvature flow Laplace Beltrami operator in a diffusion equation.

The Catmull-Clark subdivision surfaces together with Shape Enhancement can easily generate families of shapes by changing a single parameter.

Options



Repeat: 1

Axis: ✓ X ✓ Y ✓ Z

Lambda: Factor: 0.000010 Border: 0.000050

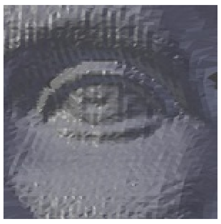
✓ Preserve Volume

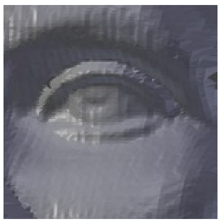
✓ Normalized Version

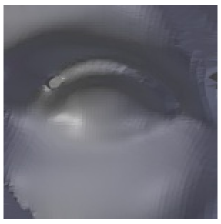
Laplacian Smooth and Shape Enhanced modifier

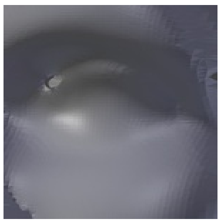
Repeat
: Repetitions allow you to run the Laplacian smoothing and Shape Enhancement multiple times. Each repetition causes the flow curvature of the mesh to be recalculated again, and as a result it removes more noise at every new iteration in Laplacian smoothing cases (positive factor) using a small Factor < **1.0**. In a Shape Enhancement case (negative Factor) multiple iterations can magnify the noise.

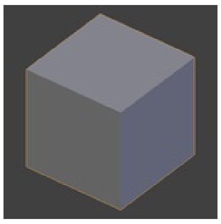
- Repeat: **0** Disables the modifier and no repetition is made.
- Repeat: **1** to **200** Number of repetitions to be done by the modifier. Be careful with large numbers of vertices, because it will take a lot of time to execute all iterations.

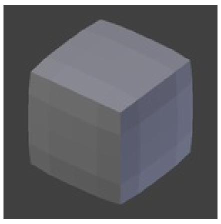

Repeat: 0, Lambda_Factor: 0.5

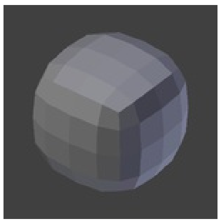

Repeat: 1, Lambda_Factor: 0.5

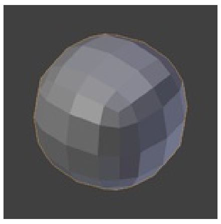

Repeat: 5, Lambda_Factor: 0.5

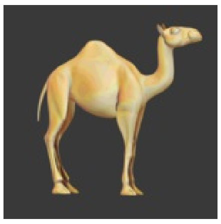

Repeat: 10, Lambda_Factor: 0.5

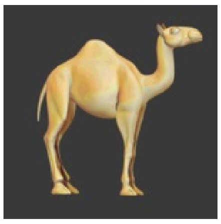

Repeat: 0, Lambda_Factor: 2.0

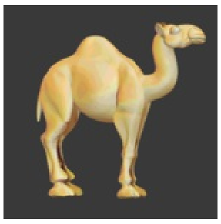

Repeat: 1, Lambda_Factor: 2.0

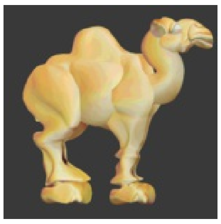

Repeat: 5, Lambda_Factor: 2.0


Repeat: 10, Lambda_Factor: 2.0


Repeat: 0, Lambda_Factor: -0.5

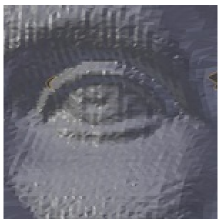

Repeat: 1, Lambda_Factor: -0.5

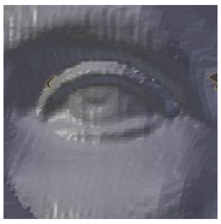

Repeat: 5, Lambda_Factor: -0.5

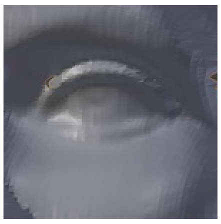

Repeat: 10, Lambda_Factor: -0.5

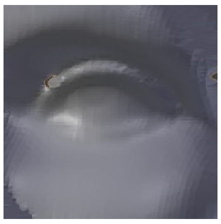
Lambda factor
: The Lambda factor ranges from **-1000.0** to **1000.0**; this factor controls the amount of displacement of every vertex along the curvature flow.

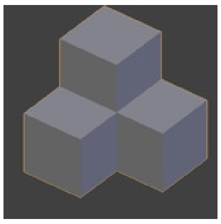
- Lambda factor: **-1000.0** to **0.0** Using a Lambda factor you can enhance the shape, preserving desirable geometry.
- Lambda factor: **0.0** Disables the modifier and no smoothing or enhancing is done.
- Lambda factor: **0.0** to **5.0** Using a small Lambda factor, you can remove noise from the shape without affecting desirable geometry.
- Lambda factor: **5.0** to **1000.0** Using a large Lambda factor you get smoothed versions of the shape at the cost of losing fine geometry details.

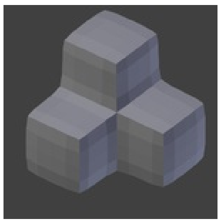

Repeat: 1, **Lambda_Factor: 0.0**

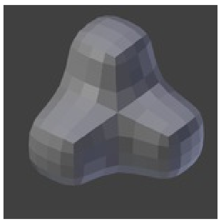

Repeat: 1, **Lambda_Factor: 0.5**

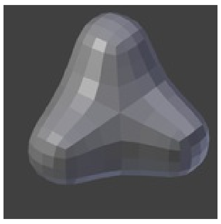

Repeat: 1, **Lambda_Factor: 2.5**

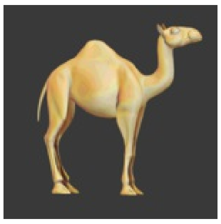

Repeat: 1, **Lambda_Factor: 5.0**

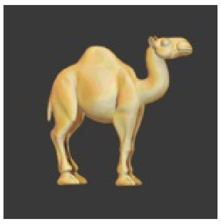

Repeat: 1, **Lambda_Factor: 0.0**

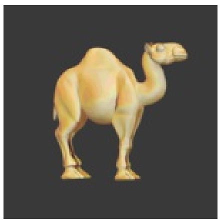

Repeat: 1, **Lambda_Factor: 1.0**

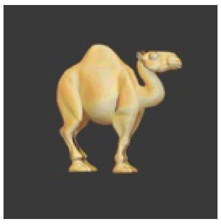

Repeat: 1, **Lambda_Factor: 10.0**


Repeat: 1, **Lambda_Factor: 50.0**


Repeat: 1, **Lambda_Factor: 0.0**

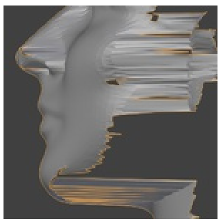

Repeat: 1, **Lambda_Factor: -20.0**

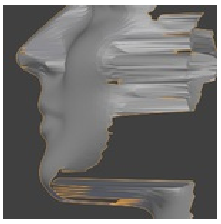

Repeat: 1, **Lambda_Factor: -50.0**

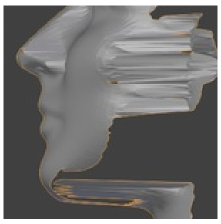

Repeat: 1, **Lambda_Factor: -300.0**

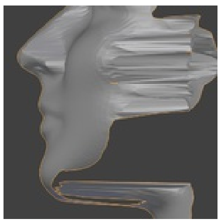
Lambda border
: The Lambda border ranges from **-1000.0** to **1000.0** . Borders are treated differently. There is no way to calculate the curvature flow on them. For this reason the Lambda factor just smooths or enhances them.

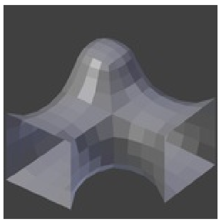
- Lambda border: **-1000.0** to **0.0** Enhance the borders.
- Lambda border: **0.0** Disables the modifier and no smoothing on the borders is done.
- Lambda border: **0.0** to **10.0** Smooths the borders.
- Lambda border: **10.0** to **1000.0** Collapses the borders in a small circle.

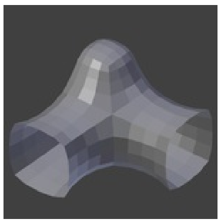

Repeat: 1, Lambda_Factor: 2.5, **Lambda_Border: 0.0**

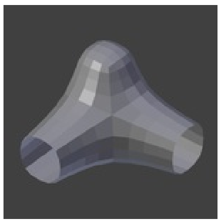

Repeat: 1, Lambda_Factor: 2.5, **Lambda_Border: 1.0**

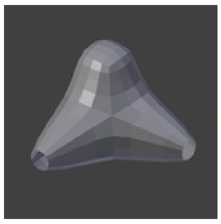

Repeat: 1, Lambda_Factor: 2.5, **Lambda_Border: 2.5**

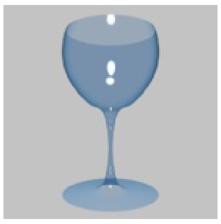

Repeat: 1, Lambda_Factor: 2.5, **Lambda_Border: 10.0**

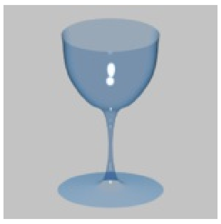

Repeat: 1, Lambda_Factor: 20.0, **Lambda_Border: 0.0**

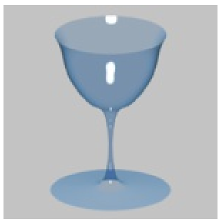

Repeat: 1, Lambda_Factor: 20.0, **Lambda_Border: 1.0**

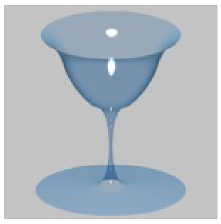

Repeat: 1, Lambda_Factor: 20.0, **Lambda_Border: 5.0**


Repeat: 1, Lambda_Factor: 20.0, **Lambda_Border: 20.0**

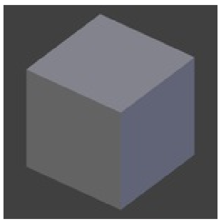

Repeat: 1, Lambda_Factor: -30.0, **Lambda_Border: 0.0**

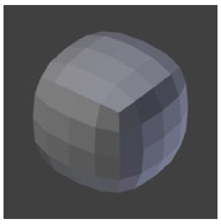

Repeat: 1, Lambda_Factor: -30.0, **Lambda_Border: -20.0**

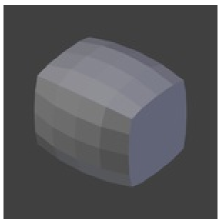

Repeat: 1, Lambda_Factor: -30.0, **Lambda_Border: -50.0**

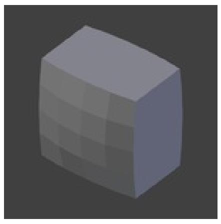

Repeat: 1, Lambda_Factor: -30.0, **Lambda_Border: -200.0**

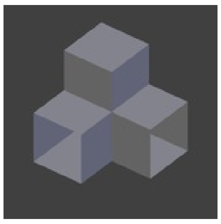
X, Y, Z
: Toggle buttons to enable/disable hard constraints in the X, Y and/or Z axis directions.

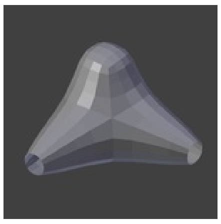

Repeat: 1, Lambda_Factor: 40.0, **X, Y, Z: Unselected**

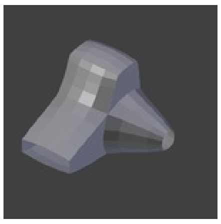

Repeat: 1, Lambda_Factor: 40.0, **X, Y, Z: Selected**

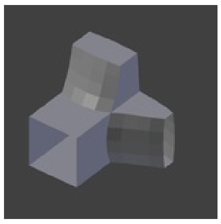

Repeat: 1, Lambda_Factor: 40.0, **X, Y: Selected**


Repeat: 1, Lambda_Factor: 40.0, **X: Selected**

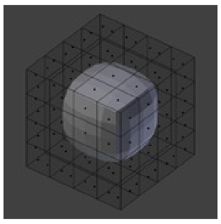

Repeat: 1, Lambda_Factor: 20.0, **X, Y, Z: Unselected**

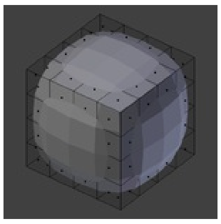

Repeat: 1, Lambda_Factor: 20.0, **X, Y, Z: Selected**

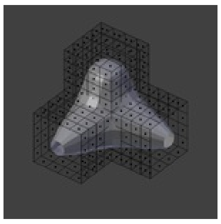

Repeat: 1, Lambda_Factor: 20.0, **X, Y: Selected**

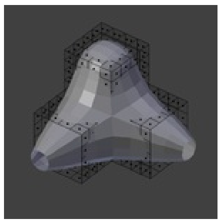

Repeat: 1, Lambda_Factor: 20.0, **X: Selected**

Preserve Volume
: The smoothing process can produce shrinkage. That is significant for large Lambda factor or large Repeat values; to reduce that effect you can use this option.

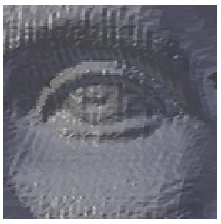

Repeat: 1, Lambda_Factor: 40.0, **Volume Preservation: Unselected**

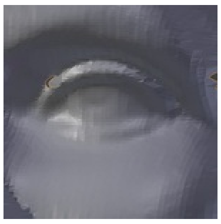

Repeat: 1, Lambda_Factor: 40.0, **Volume Preservation: Selected**

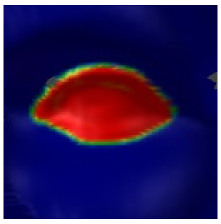

Repeat: 1, Lambda_Factor: 20.0, **Volume Preservation: Unselected**

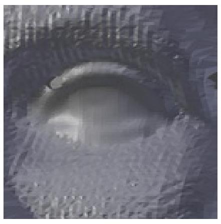

Repeat: 1, Lambda_Factor: 20.0, **Volume Preservation: Selected**

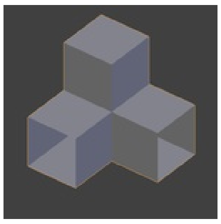
Vertex Group
: A vertex group name, to constrain the effect to a group of vertices only. Allows for selective, real-time smoothing or enhancing, by painting vertex weights.

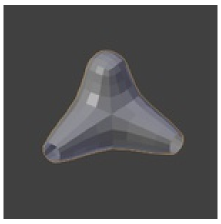

Repeat: 1, Lambda_Factor: 0.0

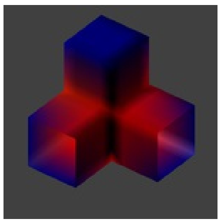

Repeat: 1, Lambda_Factor: 2.5

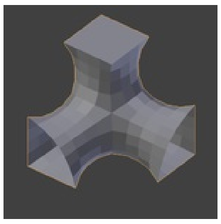

Weight Paint, Vertex Group: Group

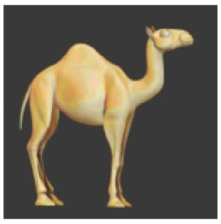

Repeat: 1, Lambda_Factor: 2.5, **Vertex Group: Group**

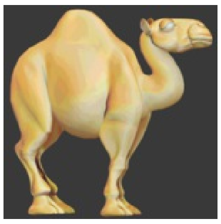

Repeat: 1, Lambda_Factor: 0.0

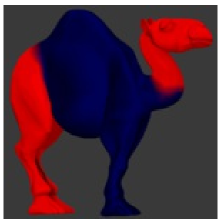

Repeat: 1, Lambda_Factor: 20.0

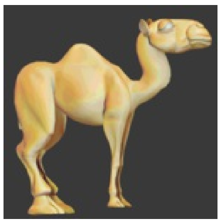

Weight Paint, Vertex Group: Group


Repeat: 1, Lambda_Factor: 20.0, **Vertex Group: Group**

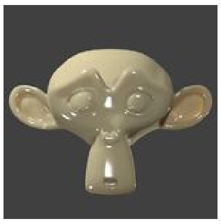

Repeat: 1, Lambda_Factor: 0.0

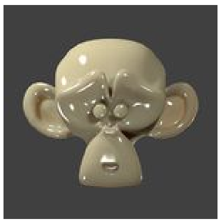

Repeat: 1, Lambda_Factor: -240.0

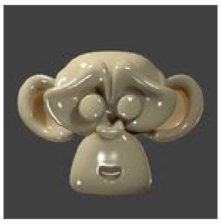

Weight Paint, Vertex Group: Group

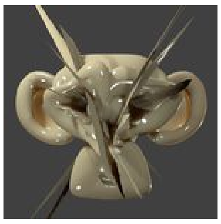

Repeat: 1, Lambda_Factor: -240.0, **Vertex Group: Group**

Normalized Version
: The modifier has two versions, the normalized version that does not depend on face size, and the other that is dependent on the face size. Be careful with the face-size-dependent version, which can produce peaks.


Normalized Version: Selected, Lambda_Factor: 0.0


Normalized Version: Selected, Lambda_Factor: -50

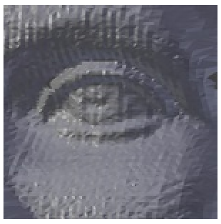

Normalized Version: Unselected, Lambda_Factor: -50

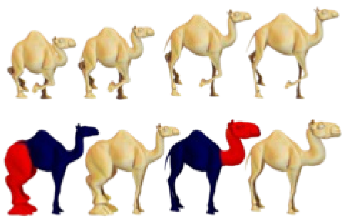

Normalized Version: Unselected, Lambda_Factor: -250

Hints

Meshes with a great number of vertices, more than ten thousand (10,000), may take several minutes for processing; you can use small portions of the mesh for testing before executing the modifier on the entire model.

Examples


Femme front view
Femme Front blend file


Camel Enhanced
Cube Smooth blend file

See Also

- Smooth Modifier

Laplacian Deform

Mode: Any mode

Panel: Modifiers

Description

The Laplacian Deform modifier allows you to pose a mesh while preserving geometric details of the surface.

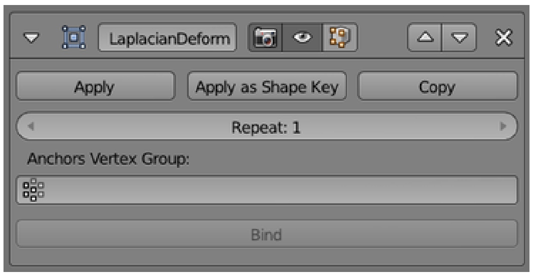
The user defines a set of 'anchor' vertices, and then moves some of them around. The modifier keeps the rest of the anchor vertices in fixed positions, and calculates the best possible locations of all the remaining vertices to preserve the original geometric details.

This modifier captures the geometric details with the uses of differential coordinates. The differential coordinates captures the local geometric information how curvature and direction of a vertex based on its neighbors.

Note
You must define a Anchors Vertex Group. Without a vertex group modifier does nothing.

Options

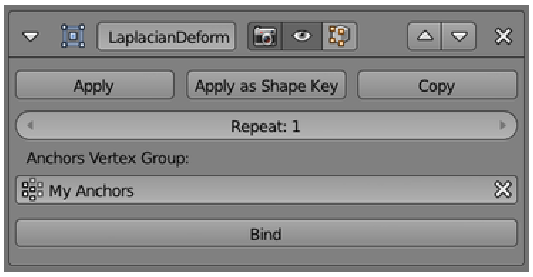
Anchors Vertex Group
: A vertex group name, to define the group of vertices that user uses to transform the model. The weight of each vertex does not affect the behavior of the modifier; the method only takes into account vertices with weight greater than 0.



Repeat: 1
Anchors Vertex Group:

Bind button is disabled until the user select a valid Anchors Vertex Group

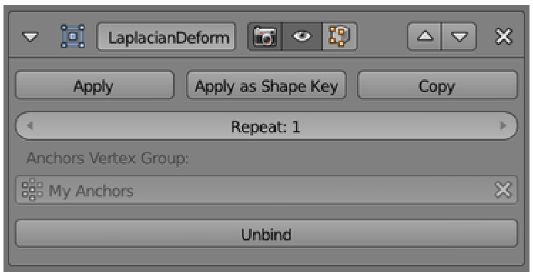
Bind
: The Bind button is what tells the Laplacian Deform modifier to actually capture the geometry details of the object, so that altering the anchors vertexes actually alters the shape of the deformed object.



Repeat: 1
Anchors Vertex Group: My Anchors

Bind button is enabled when the user selected a valid Anchors Vertex Group

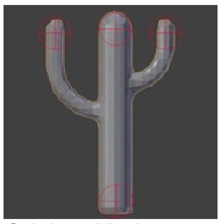
Unbind
: When a geometry details has been captured from a mesh, it can later be released by selecting the Unbind button which replaced the Bind one.

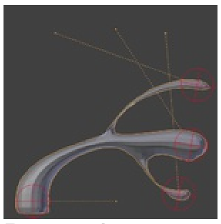


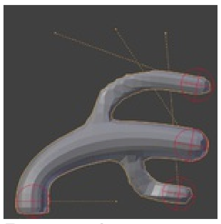
Repeat: 1
Anchors Vertex Group: My Anchors

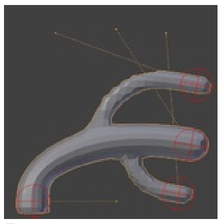
Unbind button is enabled after the user pressed the Bind button

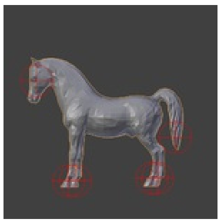
Repeat
: Repetitions iteratively improve the solution found. The objective of the system is to find the rotation of the differential coordinates preserving the best possible geometric detail. The system retains details better if more repetitions are used. A small Repeat number is recommended, as the system takes a long time to calculate each repetition.

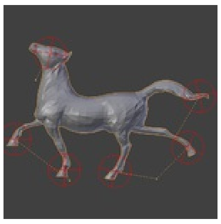

Original Model

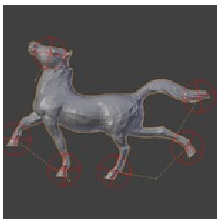

Repeat: 1

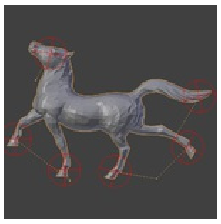

Repeat: 2


Repeat: 5


Original Model


Repeat: 1

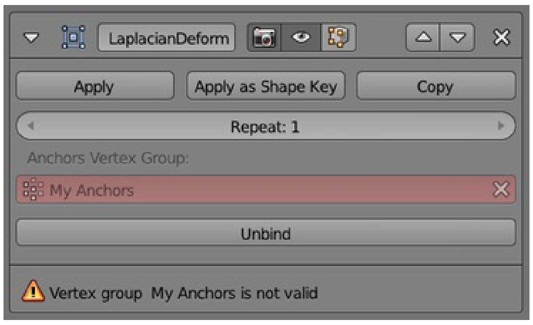

Repeat: 2


Repeat: 10

Hints

If the mesh is dense, with a number of vertices greater than 100,000, then it is possible that the nonlinear optimization system will fail.

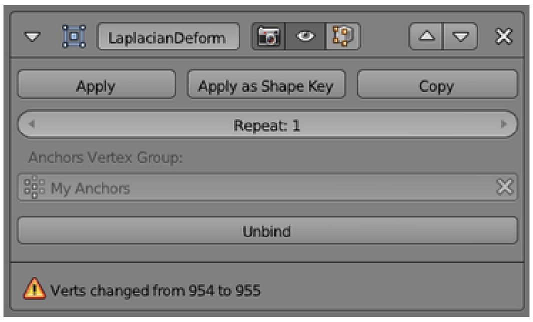
Vertex group My Anchors is not valid
: This message is displayed when a user deletes a Vertex Group or when the user changes the name of the Vertex Group.



Repeat: 1
Anchors Vertex Group:
My Anchors
Vertex group My Anchors is not valid

My Anchors is the anchors vertex group for this example

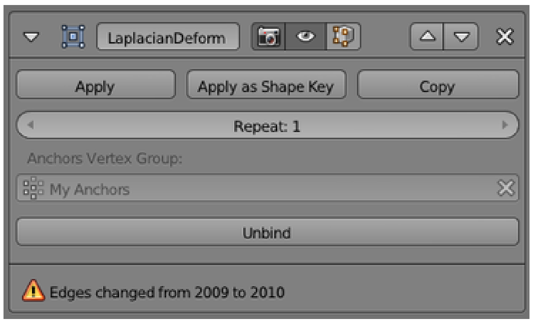
Verts changed from 954 to 955
: This message is displayed when a user add or delete verts to the mesh.



Repeat: 1
Anchors Vertex Group:
My Anchors
Verts changed from 954 to 955

954 to 955 correspond to the number of verts changed by user before and after

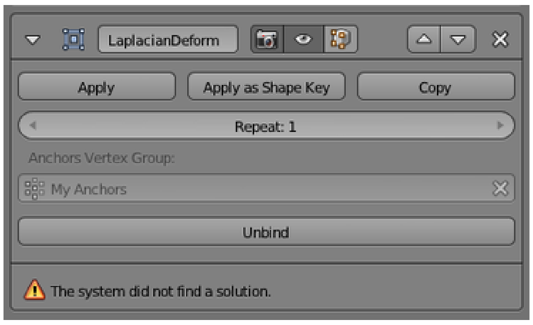
Edges changed from 2009 to 2010
: This message is displayed when a user add or delete edges to the mesh.



Repeat: 1
Anchors Vertex Group:
My Anchors
Edges changed from 2009 to 2010

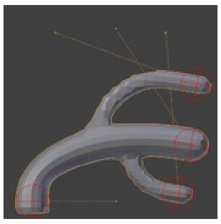
2009 to 2010 correspond to the number of edges changed by user before and after

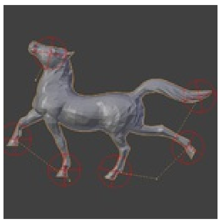
The system did not find a solution
: This message is displayed if the solver SuperLU did not find a solution for the linear system.



Repeat: 1
Anchors Vertex Group:
My Anchors
The system did not find a solution.

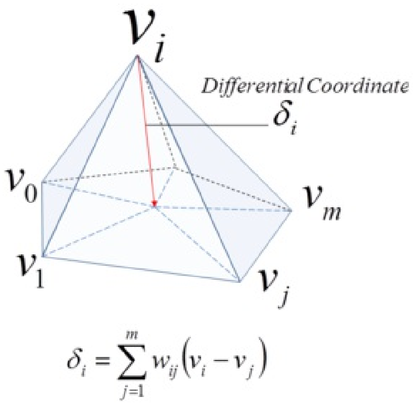
Examples


Cactus example with Armature object
Download Cactus blend file


Horse example with Hook objects
Download Horse blend file

History

Laplacian Surface Editing is a method developed by Olga Sorkine and others in 2004. This method preserves geometric details as much as possible while the user makes editing operations. This method uses differential coordinates corresponding to the difference between a vector and the weighted average of its neighbors to represent the local geometric detail of the mesh.



Differential Coordinate

See Also

Laplacian Surface Editing (Original paper)

Differential Coordinates for Interactive Mesh Editing

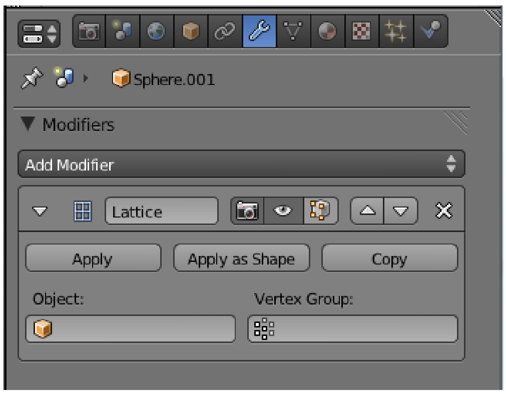
Lattice Modifier

Mode: Object mode

Panel: Properties Window -> Context Button Modifiers

The Lattice modifier deforms the base object according to the shape of a Lattice object.

Options



Lattice modifier.

Object
: The Lattice object with which to deform the base object.

Vertex Group
: An optional vertex group name which lets you limit the modifier effect to a part of the base mesh.

Method

You can control the Lattice object attributes in the Object Data context for the Lattice in the Properties Window .

A lattice consists of a non-renderable three-dimensional grid of vertices. Its main use is to give extra deformation capabilities to the underlying object it controls (either *via* a modifier, or as its parent). Objects to be deformed can be meshes, curves, surfaces, text, lattices and even particles.

The lattice should be scaled and moved to fit around your object in object mode. Any scaling applied to the object in edit mode will result in the object deforming. This includes applying scale with CtrlA as this will achieve the same result as scaling the lattice in edit mode, and therefore the object.

Hints

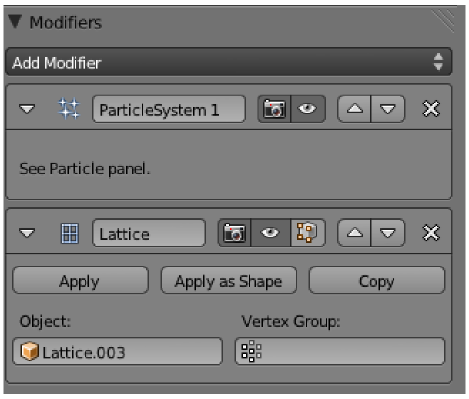
Why would you use a lattice to deform a mesh instead of deforming the mesh itself in Edit mode? There are a couple of reasons for that:

- First of all: it’s easier. Since your mesh could have a zillion vertices, scaling, grabbing and moving them could be a hard task. Instead, if you use a nice simple lattice your job is simplified to moving just a couple of vertices.
- It’s nicer. The deformation you get looks a lot better!
- It’s fast! You can use the same lattice to deform several meshes. Just give each object a lattice modifier, all pointing to the same lattice.
- It’s a good practice. A lattice can be used to get different versions of a mesh with minimal extra work and consumption of resources. This leads to an optimal scene design, minimizing the amount of modeling work. A lattice does not affect the texture coordinates of a mesh’s surface. Subtle changes to mesh objects are easily facilitated in this way, and do not change the mesh itself.

Example/Tutorial(s)

There are example tutorials for 2.4 versions in the Tutorials section. A 2.6 tutorial shows how to shape a fork.

Particles and Lattices



Particles following a lattice.

Particles follow a Lattice if the modifier sequence is right. First the particles, then the lattice!

Mesh Deform Modifier

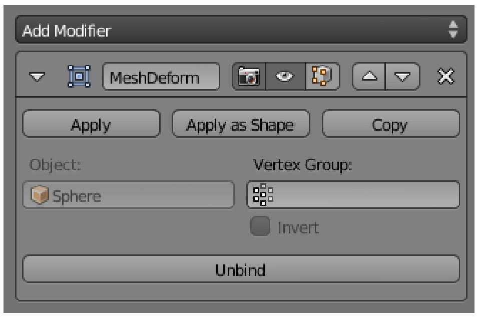
Mode: All modes

Panel: Modifiers

Description

The Mesh Deform modifier allows an arbitrary closed mesh (of any closed shape, not just the cuboid shape of a Lattice modifier) to act as a deformation cage around another mesh.

Options



Mesh Deform modifier

The Mesh Deform modifier is reasonably easy to use but it can be very slow to do the calculations it needs, to properly map the deform mesh cage to the deformed object.

Object
: The name of the mesh object to be used as a deforming mesh cage.

Vertex Group
: An optional vertex group that will be affected by the deforming mesh cage.

Invert
: Inverts the influence set by the vertex group defined in previous setting (i.e. reverts the weight values of this group).

Bind
: The Bind button is what tells the Mesh Deform modifier to actually link the deform mesh cage to the deformed object, so that altering the shape of the deform mesh cage actually alters the shape of the deformed object.
Be aware that depending on the settings of the Mesh Deform modifier and complexity of the deform mesh cage and/or deformed object, it can take a long time for this operation to complete. This can result in Blender not responding to user's actions until it has completed, it is even possible that Blender will run out of memory and crash.

Unbind
: When a deformed object has been associated to a deform mesh cage, it can later be disassociated by selecting the Unbind button which replaced the Bind one.
When Unbind is clicked, the *deform mesh cage* will keep its current shape; it will not reset itself back to its original start shape. If you need its original shape, you will have to save a copy of it before you alter it. The deformed object will, however, reset back to its original shape that it had before it was bound to the deform mesh cage.

Precision
: The Precision numeric slider field controls the accuracy with which the deform mesh cage alters the deformed object, when the points on the cage are moved.
The range of values for the Precision field can range from **2** to **10**, the default being **5**. Raising this value higher can greatly increase the time it takes the Mesh Deform modifier to complete its binding calculations, but it will get more accurate cage mapping to the deformed object. This rise in calculation time can make Blender stop responding until it has calculated what it needs to. As well as making Blender not respond, raising the Precision value high and then trying to Bind on a very complex deform mesh cage and/or deformed object can use large amounts of memory and in extreme cases crash Blender. To be safe, save your blend file before proceeding!
This setting becomes unavailable once a cage has been bound.

Dynamic
: The Dynamic button indicates to the Mesh Deform modifier that it should also take into account deformations and changes to the underlying deformed object which were not a direct result of deform mesh cage alteration.
With the Dynamic button activated, other mesh altering features (such as other modifiers and shape keys) are taken into account when binding a deform mesh cage to the deformed object, increasing deformation quality. It is deactivated by default to save memory and processing time when binding…
Like with Precision, this setting is unavailable once a cage has been bound.

Hints

- Ensure that the normals on the cage mesh point to the outside; they are used to determine the inside and outside of the cage.
- Besides the outer cage, more faces within the cage, either loose or forming another smaller cage, can be used for extra control. Such smaller cages may also overlap with the main cage; for example, to get extra control over eyes, two small sphere cages

could be added around them.

See Also

- The Lattice modifier.
- [http://graphics.pixar.com/library/HarmonicCoordinatesB/ (original paper)

Shrinkwrap Modifier

Mode: All modes

Panel: Modifiers

Description

The Shrinkwrap modifier allows an object to “shrink” to the surface of another object. It moves each vertex of the object being modified to the closest position on the surface of the given mesh (using one of the three methods available). It can be applied to meshes, lattices, curves, surfaces and texts.

Like most of the deform modifiers, the affected “vertices” are the “computed” one, i.e. the real geometry of the object at the time the modifier is calculated, and not the original *vertices*/control points.

Something of a view-independent retopo tool (in Blender 2.49), Shrinkwrap projects vertices along their normals or moved to the nearest surface point. But it doesn’t have accuracy problems like retopo did, since it works in object space instead of image space. Also it’s possible to “keep a distance” from the target position.

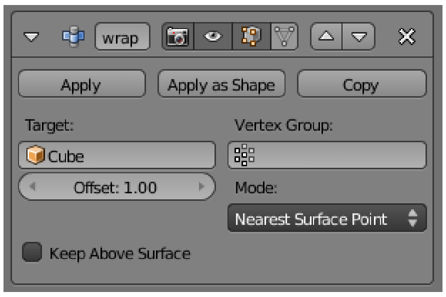
Note
For those who found the Shrinkwrap modifier pretty useful, but would like it to move empties or object's positions … have a look at the Shrinkwrap constraint!

Options

Target
: Shrink target, the mesh to shrink/wrap around.

Vertex Group
: The weight paint for this vertex group of the current modified mesh controls whether and how much each vertex is displaced to its target position. If a vertex is not a member of this group, it is not displaced (same as weight 0).

Offset
: The distance that must be kept from the calculated target position, in Blender Units.

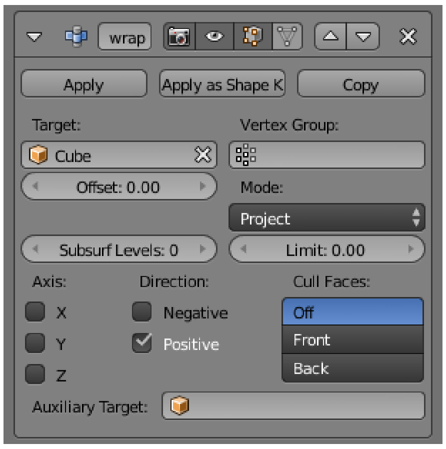


Target: Cube
Vertex Group:
Offset: 1.00
Mode: Nearest Surface Point

Nearest Surface Point

Mode
: This drop-down list specifies the method to be used to determine the nearest point on the target’s surface for each vertex of the modified object. Some options will add some extra, specific controls to the panel.

    Nearest Surface Point
    : This will select the nearest point over the surface of the shrink target. It adds the extra option Above surface, which always keep the computed vertices above their "floor faces". This is only meaningful when Offset is not null.



Target: Cube
Vertex Group:
Offset: 0.00
Mode: Project
Subsurf Levels: 0
Limit: 0.00
Direction: Positive
Cull Faces: Off
Auxiliary Target:

Project

Projection
: This will project vertices along a chosen axis until they touch the shrink target. Vertices that never touch the shrink target are left in their original position. This implies that, depending on the settings of this option and the relative positions of the two objects, the modified object might sometimes remain undeformed. This is not a bug; just “play” with the settings (especially the Negative/Positive ones), or move one of the objects around…

[illegible]
This method is the hardest to master, as it might sometimes give unexpected results… It adds quite a few extra options:

Subsurf Levels

This applies a (temporary) Catmull-Clark subsurf to the modified object, before computing the wrap when using Projection mode.

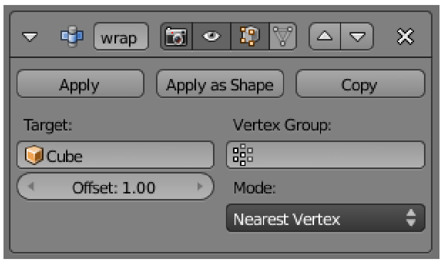
Subsurf Limit
: This is a distance limit between original vertex and surface. If the distance is larger than this limit vertex wouldn't be projected onto the surface,

X, Y, Z
: Along which local axis of the modified object the projection is done. These options can be combined with each other, yielding a “median axis” of projection.

Negative, Positive
: This allows you to select the allowed direction(s) of the shrink along the selected axis. With more than one Shrinkwrap modifier, negative and positive axes can be combined.

Cull Faces
: This allows you to prevent any projection over the “front side” (respectively the “back side”) of the target’s faces. The “side” of a face is determined by its normal (front being the side “from where” the normal “originates”).

Auxiliary Target
: An additional object to project over.


wrap

Apply | Apply as Shape | Copy

Target: Cube
Offset: 1.00
Vertex Group:
Mode: Nearest Vertex

Nearest Vertex

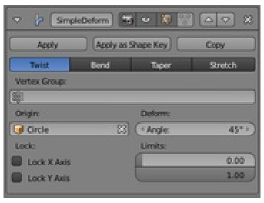
Nearest Vertex
: This will select the nearest vertex of the shrink target. It adds no extra options.

Simple Deform Modifier

Mode: All modes

Panel: Modifiers

Description



Simple Deform

The Simple Deform modifier allows easy application of a simple deformation to an object (meshes, lattices, curves, surfaces and texts are supported).

Like most of the deform modifiers, the deform functions are applied to the "computed vertices", i.e. to the real geometry of the object at the time the modifier is calculated, and not to the original *vertices*/control points. This means you can increase the level of detail of the deform effect by first inserting a Subdivision Surface modifier (for meshes), or raising the Preview Resolution settings (for curves/surfaces/texts).

Using another object, it's possible to define the axis and origin of the deformation, allowing application of very different effects.

Options

Mode
: This drop-down list defines the deform function applied, among four available:

- Twist – Rotates around the Z axis.
- Bend – Bends the mesh over the Z axis.
- Taper – Linearly scales along Z axis.
- Stretch – Stretches the object along the Z axis (negative Factor leads to squash).

Vertex Group
: The name of the vertex group that indicates whether and how much each vertex is influenced by the deformation.

Origin
: The name of an object that defines the origin of deformation (usually an empty). This object can be:

- Rotated to control the axis (as it is its Z axis that is now used as "guide").
- Translated to control the origin of deformation.
- Scaled to change the deform factor.

Note
When the object controlling the origin (the one in the Origin field) is a child of the deformed object, this creates a cyclic dependency in Blender's data system (the DAG – "dependency graph"?). The workaround is to create a new empty and attach both objects to it.

Factor
: The amount of deformation. Can be set to negative.

Limits
: These settings allow you to set the lower and upper limits of the deformation (they are proportional values, from **0.0** to **1.0**). Obviously, the upper limit can't be lower than lower limit.

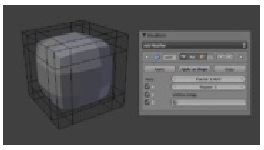
Lock X Axis/Lock Y Axis (Taper and Stretch modes only)
: These controls whether the X and/or Y coordinates are allowed to change or not. Thus it is possible to squash the X coordinates of an object and keep the Y coordinates intact.

Smooth Modifier

Mode: Any mode

Panel: Modifiers

Description


Smooth modifier applied to a subdivided cube

As above, with a vertex group selected

This modifier smooths a mesh by flattening the angles between adjacent faces in it, just like Smooth in the Editing context. It smooths without subdividing the mesh – the number of vertices remains the same.

This modifier is not limited to smoothing, though. Its control factor can be configured outside the [**0.0**, **1.0**] range (including negative values), which can result in interesting deformations, depending on the affected mesh.

Options

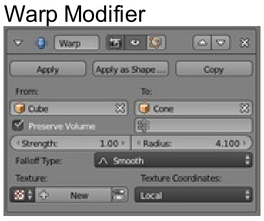
X, Y, Z
: Toggle buttons to enable/disable the modifier in the X, Y and/or Z axes directions.

Factor
: The factor to control the smoothing amount. The smoothing ranges from **0.0** to **1.0** (**0.0**: disabled, **0.5**: same as the Smooth button, **1.0**: maximum). Alternatively, values outside this range (above **1.0** or below **0.0**) distort the mesh.

Repeat
: The number of smoothing iterations, equivalent to pressing the Smooth button multiple times.

Vertex Group
: A vertex group name, to restrict the effect to the vertices in it only. This allows for selective, real-time smoothing, by painting vertex weights.

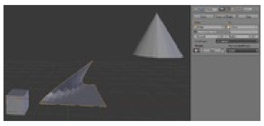
Warp Modifier



From: Cube
To: Cone
Preserve Volume ✓
Strength: 1.00
Radius: 4.100
Falloff Type: Smooth
Texture: New
Texture Coordinates: Local

Warp modifier

This deformation modifier can be used to warp parts of a mesh to a new location in a very flexible way by using 2 objects to select the "from" and "to" regions, with options for using a curve falloff, texture and vertex group.


Warp modifier applied to a grid

The Warp Modifier is a bit tricky at first, but it helps to understand how it works. The modifier requires two points, specified by object centers. The "from" point designates a point in space that is pulled toward the "to" point. It is akin to using the Proportional Editing in edit mode.

Options

From:
: Specify the origin object transformation of the warp.

To:
: Specify the destination object transformation of the warp.

Preserve Volume
: Enables volume preservation when rotating one of the transforms.

Vertex Group
: Limit the deformation to a specific vertex group.

Strength
: Sets how strong the effect is.

Radius
: Sets the distance from the transforms that can be warped by the transform handles.

Falloff Type
: Sets the way the strength of the warp change as it goes from the center of the transform to the Radius value. See Proportional Editing for descriptions of the falloff types.

Texture
: Specify a texture the strength is offset by to create variations in the displacement.

Texture Coordinates
: Set the way textures are applied to the mesh when using a textured warp.

    Object
    : Specify an object to use when set to Object.

    UV Layer
    : Specify a UV layer when set to UV.

Wave Modifier

Mode: Object mode

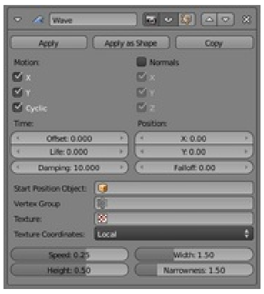
Panel: Modifiers

Description

The Wave modifier adds an ocean-like motion to the Z coordinate of the object's vertices/control points. This modifier is available for meshes, lattices, curves, surfaces and texts, with a few restrictions for non-*mesh* objects:

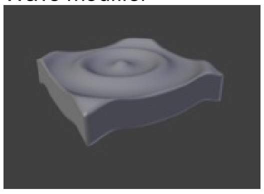
- Activating Normals or typing a name in VGroup will simply deactivate the modifier.
- Even worse, selecting UV as texture coordinates will make Blender crash at once!

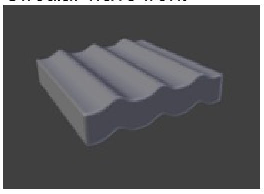
Options

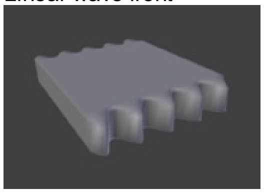


Motion: [✓] X [✓] Y [✓] Cyclic
Normals: [ ] Normals, [✓] X [✓] Y [✓] Z
Time: Offset: 0.000, Life: 0.000, Damping: 10.000
Position: X: 0.00, Y: 0.00, Falloff: 0.00
Start Position Object:
Vertex Group
Texture:
Texture Coordinates: Local
Speed: 0.25, Height: 0.50, Width: 1.50, Narrowness: 1.50

Wave modifier


Circular wave front


Linear wave front


Motion enabled for X and Normals enabled for Y

Motion
: X, Y, Cyclic: The wave effect deforms vertices/control points in the Z direction, originating from the given starting point and propagating along the object with circular wave fronts (both X and Y activated), or with rectilinear wave fronts, then parallel to the axis corresponding to the X or Y button activated. Cyclic repeats the waves cyclically, rather than a single pulse.

Normals
: For meshes only. Displaces the mesh along the surface normals (instead of the object's Z-axis).

Time
: Settings to control time parameters.

  Offset
  : Time offset in frames. The frame at which the wave begins (if Speed is positive), or ends (if Speed is negative). Use a negative frame number to prime and pre-start the waves.

  Life
  : Duration of animation in frames. Set to zero, loops the animation forever.

  Damping
  : An additional number of frames in which the wave slowly damps from the Height value to zero after Life is reached. The dampening occurs for all the ripples and begins in the first frame after the Life is over. Ripples disappear over Damping frames.

Position
: X and Y coordinates of the center of the waves, in the object's local coordinates. Falloff controls how fast the waves fade out as they travel away from the coordinates above. Note that selecting a Start Position Object effectively cancels the coordinates chosen above, but retains the Falloff value.

Start Position Object
: Use another object as the reference for the starting position of the wave. Leave blank to disable. Note that you then can animate this object’s position, to change the wave's origin across time.

Vertex Group
: For meshes only. A vertex group name, used to control the parts of the mesh affected by the wave effect, and to what extent (using vertex weights).

Texture
: Use this texture to control the object’s displacement level. Animated textures can give very interesting results here.

Texture Coordinates
: This menu lets you choose the texture’s coordinates for displacement:

  Local
  : Object's local coordinates.

  Global
  : Global coordinates.

  Object
  : Adds an additional field just below, to type in the name of the object from which to get the texture coordinates.

  UV
  : Adds an extra UV Layer drop-down list, to select the UV layer to be used. **Warning:** do not activate this option with non-mesh objects; it seems to make Blender crash.

Speed
: The speed, in BU (for “Blender Units”) per frame, of the ripple.

Height
: The height or amplitude, in BU, of the ripple.

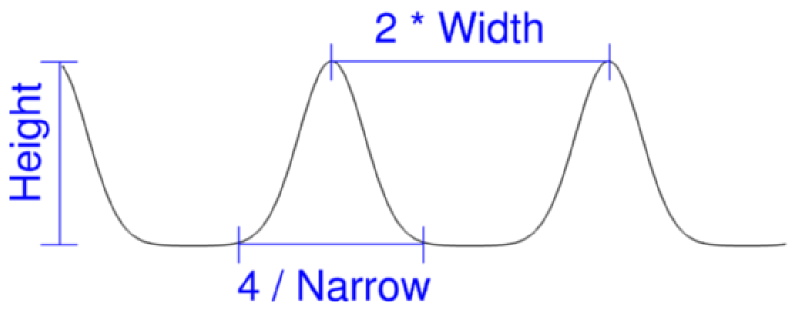
Width
: Half of the width, in BU, between the tops of two subsequent ripples (if Cycl is enabled). This has an indirect effect on the ripple amplitude – if the pulses are too near to each other, the wave may not reach the **0** Z-position, so in this case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the maximum is lower than the expected amplitude. See technical details below.

Narrowness
: The actual width of each pulse: the higher the value the narrower the pulse. The actual width of the area in which the single pulse is apparent is given by `4/Narrowness`. That is, if Narrowness is **1** the pulse is **4** units wide, and if Narrowness is **4** the pulse is **1** unit wide.

Warning
All the values described above must be multiplied with the corresponding Scale values of the object to get the real dimensions. For example, if the value of Scale Z is **2** and the value of Height of the waves is **1**, it gives us final waves with a height of **2 BU**!

Technical Details and Hints

The relationship of the above values is described here:

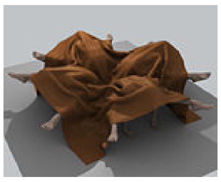


Wave front characteristics.

To obtain a nice wave effect similar to sea waves and close to a sinusoidal wave, make the distance between following ripples and the ripple width equal; that is, the Narrowness value must be equal to `2/Width`. E.g. for Width=**1**, set Narrow to **2**.

Cloth Simulation

Cloth example.


Cloth on carved wooden men (made by motorsep).

Cloth example.

Cloth simulation is one of the hardest aspects of CG, because it is a deceptively simple real-world item that is taken for granted, yet actually has very complex internal and environmental interactions. After years of development, Blender has a very robust cloth simulator that is used to make clothing, flags, banners, and so on. Cloth interacts with and is affected by other moving objects, the wind and other forces, as well as a general aerodynamic model, all of which is under your control.

Description

A piece of cloth is any mesh, open or enclosed, that has been designated as cloth. The Cloth panels are located in the Physics sub-context and consist of three panels of options. Cloth is either an open or closed mesh and is mass-less, in that all cloth is assumed to have the same density, or mass per square unit.

Cloth is commonly modeled as a mesh grid primitive, or a cube, but can also be, for example, a teddy bear. However, Blender's Softbody system provides better simulation of closed meshes; Cloth is a specialized simulation of fabrics.

Once the object is designated as Cloth, a Cloth modifier will be added to the object's modifier stack automatically. As a modifier then, it can interact with other modifiers, such as Armature and Smooth. In these cases, the ultimate shape of the mesh is computed in accordance with the order of the modifier stack. For example, you should smooth the cloth *after* the modifier computes the shape of the cloth.

So you edit the Cloth settings in two places: use the F7 Physics buttons to edit the properties of the cloth and use the Modifier stack to edit the Modifier properties related to display and interaction with other modifiers.

You can Apply the cloth modifier to freeze, or lock in, the shape of the mesh at that frame, which removes the modifier. For example, you can drape a flat cloth over a table, let the simulation run, and then apply the modifier. In this sense, you are using the simulator to save yourself a lot of modeling time.

Results of the simulation are saved in a cache, so that the shape of the mesh, once calculated for a frame in an animation, does not have to be recomputed again. If changes to the simulation are made, you have full control over clearing the cache and re-running the simulation. Running the simulation for the first time is fully automatic and no baking or separate step interrupts the workflow.

Computation of the shape of the cloth at every frame is automatic and done in the background; thus you can continue working while the simulation is computed. However it is CPU-intensive and depending on the power of your PC and the complexity of the simulation, the amount of CPU needed to compute the mesh varies, as does the lag you might notice.

Don't jump ahead
If you set up a cloth simulation but Blender has not computed the shapes for the duration of the simulation, and if you jump ahead a lot of frames forward in your animation, the cloth simulator may not be able to compute or show you an accurate mesh shape for that frame, if it has not previously computed the shape for the previous frame(s).

Workflow

A general process for working with cloth is to:

1. Model the cloth object as a general starting shape.
2. Designate the object as a "cloth" in the Physics tab of the Properties window.
3. Model other deflection objects that will interact with the cloth. Ensure the Deflection modifier is last on the modifier stack, after any other mesh deforming modifiers.
4. Light the cloth and assign materials and textures, UV-unwrapping if desired.
5. If desired, give the object particles, such as steam coming off the surface.
6. Run the simulation and adjust Options to obtain satisfactory results. The timeline window's VCR controls are great for this step.
7. Optionally age the mesh to some point in the simulation to obtain a new default starting shape.
8. Make minor edits to the mesh on a frame-by-frame basis to correct minor tears.

Creating Cloth Simulations

This section discusses how to use those options to get the effect you want. First, enable Cloth. Set up for the kind of cloth you are simulating. You can choose one of the presets to have a starting point.

As you can see, the heavier the fabric, the more stiff it is and the less it stretches and is affected by air.

Cloth Panel

Presets
: Contains a number of preset cloth examples, and allows you to add your own.

Quality
: Set the number of simulation steps per frame. Higher values result in better quality, but is slower.

Material

Mass
: The mass of the cloth material.

Structural
: Overall stiffness of the cloth.

Bending
: Wrinkle coefficient. Higher creates more large folds.

Damping

Spring
: Damping of cloth velocity. Higher = more smooth, less jiggling.

Air
: Air normally has some thickness which slows falling things down.

Pinning

Cloth in action.

The first thing you need when pinning cloth is a Vertex Group. There are several ways of doing this including using the Weight Paint tool to paint the areas you want to pin (see the Weight paint section of the manual).

Once you have a vertex group set, things are pretty straightforward; all you have to do is press the Pinning of cloth button in the Cloth panel and select which vertex group you want to use, and the stiffness you want it at.

Stiffness
: Target position stiffness. You can leave the stiffness as it is; the default value of 1 is fine.

Collisions

In most cases, a piece of cloth does not just hang there in 3D space, it collides with other objects in the environment. To ensure proper simulation, there are several items that have to be set up and working together:

1. The Cloth object must be told to participate in Collisions.
2. Optionally (but recommended) tell the cloth to collide with itself.
3. Other objects must be visible to the Cloth object *via* shared layers.
4. The other objects must be mesh objects.
5. The other objects may move or be themselves deformed by other objects (like an armature or shape key).
6. The other mesh objects must be told to deflect the cloth object.
7. The blend file must be saved in a directory so that simulation results can be saved.
8. You then Bake the simulation. The simulator computes the shape of the cloth for a frame range.
9. You can then edit the simulation results, or make adjustments to the cloth mesh, at specific frames.
10. You can make adjustments to the environment or deforming objects, and then re-run the cloth simulation from the current frame forward.

Collision Settings

Cloth Collisions panel.

Now you must tell the Cloth object that you want it to participate in collisions. For the cloth object, locate the Cloth Collision panel, shown to the right:

Enable Collisions
: LMB click this to tell the cloth object that it needs to move out of the way.

Quality
: A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the cloth.

Distance
: As another object gets this close to it (in Blender Units), the simulation will start to push the cloth out of the way.

Repel
: Repulsion force to apply when cloth is close to colliding.

Repel Distance

Maximum distance to apply repulsion force. Must be greater than minimum distance.

Friction
: A coefficient for how slippery the cloth is when it collides with the mesh object. For example, silk has a lower coefficient of friction than cotton.

Self-collisions

Real cloth cannot permeate itself, so you normally want the cloth to self-collide.

Enable Self Collisions
: Click this to tell the cloth object that it should not penetrate itself. This adds to simulation compute time, but provides more realistic results. A flag, viewed from a distance does not need this enabled, but a close-up of a cape or blouse on a character should have this enabled.

Quality
: For higher self-collision quality just increase the Quality and more self collision layers can be solved. Just keep in mind that you need to have at least the same Collision Quality value as the Quality value.

Distance
: If you encounter problems, you could also change the Min Distance value for the self-collisions. The best value is 0.75; for fast things you better take 1.0. The value 0.5 is quite risky (most likely many penetrations) but also gives some speedup.

Regression blend file: Cloth selfcollisions.

Shared Layers

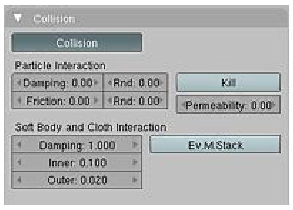
Suppose you have two objects: a pair of Pants on layers 2 and 3, and your Character mesh on layers 1 and 2. You have enabled the Pants as cloth as described above. You must now make the Character “visible” to the Cloth object, so that as your character bends its leg, it will push the cloth. This principle is the same for all simulations; simulations only interact with objects on a shared layer. In this example, both objects share layer 2.

To view/change an object’s layers, RMB click to select the object in Object mode in the 3D view. M to bring up the “Move Layers” popup, which shows you all the layers that the object is on. To put the object on a single layer, LMB click the layer button. To put the object on multiple layers, ⇧ Shift LMB the layer buttons. To remove an object from a selected layer, simply ⇧ Shift LMB the layer button again to toggle it.

Mesh Objects Collide

If your colliding object is not a mesh object, such as a NURBS surface, or text object, you must convert it to a mesh object. To do so, select the object in object mode, and in the 3D View header, select Object → Convert Object Type (AltC), and select Mesh from the popup menu.

Cloth - Object collisions



Damping: 0.00 Rnd: 0.00
Friction: 0.00 Rnd: 0.00
Permeability: 0.00
Damping: 1.000
Inner: 0.100
Outer: 0.020

Collision settings.

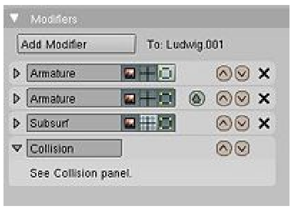
The cloth object needs to be deflected by some other object. To deflect a cloth, the object must be enabled as an object that collides with the cloth object. To enable Cloth - Object collisions, you have to enable deflections on the collision object (not on the cloth object).

In the Buttons window, Object context, Physics sub-context, locate the Collision panel shown to the right. It is also important to note that this collision panel is used to tell all simulations that this object is to participate in colliding/deflecting other objects on a shared layer (particles, soft bodies, and cloth).

Beware

There are three different Collision panels, all found in the Physics sub-context. The first (by default), a tab beside the Fields panel, is the one needed here. The second panel, a tab in the Soft Body group, concern softbodies (and so has nothing to do with cloth). And we have already seen the last one, by default a tab beside the Cloth panel.

Mesh Object Modifier Stack



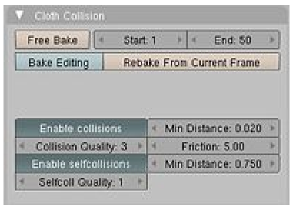
Collision stack.

The object’s shape deforms the cloth, so the cloth simulation must know the “true” shape of that mesh object at that frame. This true shape is the basis shape as modified by shape keys or armatures. Therefore, the Collision modifier must be **after** any of those. The image to the right shows the Modifiers panel for the Character mesh object (not the cloth object).

Cloth Cache

Cache settings for cloth are the same as with other dynamic systems. See Particle Cache for details.

Bake Collision



Start: 1
End: 50
Min Distance: 0.020
Collision Quality: 3
Friction: 5.00
Min Distance: 0.750
Selfcoll Quality: 1

After you have set up the deflection mesh for the frame range you intend to run the simulation (including animating that mesh *via* armatures), you can now tell the cloth simulation to compute (and avoid) collisions. Select the cloth object and in the Object context, Physics sub-context, set the Start and End settings for the simulation frames you wish to compute, and click the Bake button.

You cannot change Start or End without clearing the bake simulation. When the simulation has finished, you will notice you have the option to free the bake, edit the bake and re-bake:

There’s a few things you’ll probably notice right away. First, it will bake significantly slower than before, and it will probably clip through the box pretty badly as in the picture on the right.

Editing the cached simulation=

The cache contains the shape of the mesh at each frame. You can edit the cached simulation, after you've baked the simulation and pressed the Bake Editing button. Just go to the frame you want to fix and ⇆ Tab into Edit mode. There you can move your vertices using all of Blender’s mesh shaping tools. When you exit, the shape of the mesh will be recorded for that frame of the animation. If you want Blender to resume the simulation using the new shape going forward, LMB click ’Rebake from next Frame and play the animation. Blender will then pick up with that shape and resume the simulation.

Edit the mesh to correct minor tears and places where the colliding object has punctured the cloth.

If you add, delete, extrude, or remove vertices in the mesh, Blender will take the new mesh as the starting shape of the mesh back to the *first frame* of the animation, replacing the original shape you started with, up to the frame you were on when you edited the mesh. Therefore, if you change the content of a mesh, when you ⇆ Tab out of Edit mode, you should unprotect and clear the cache so that Blender will make a consistent simulation.

Troubleshooting

If you encounter some problems with collision detection, there are two ways to fix them:

- The fastest solution is to increase the Min Distance setting under the Cloth Collision panel. This will be the fastest way to fix the clipping; however, it will be less accurate and won’t look as good. Using this method tends to make it look like the cloth is resting on air, and gives it a very rounded look.
- A second method is to increase the Quality (in the first Cloth panel). This results in smaller steps for the simulator and therefore to a higher probability that fast-moving collisions get caught. You can also increase the Collision Quality to perform more iterations to get collisions solved.
- If none of the methods help, you can easily edit the cached/baked result in Edit mode afterwards.
- My Cloth is torn by the deforming mesh - he “Hulks Out”: Increase its structural stiffness (StructStiff setting, Cloth panel), very high, like 1000.

Subsurf modifier
A bake/cache is done for every subsurf level so please use **the equal** subsurf level for render and preview.

Examples

To start with cloth, the first thing you need, of course, is some fabric. So, let's delete the default cube and add a plane. I scaled mine up along the Y axis, but you don't have to do this. In order to get some good floppy and flexible fabric, you'll need to subdivide it several times. I did it 8 times for this example. So ⇆ Tab into Edit mode, and press W → Subdivide multi, and set it to 8.

Now, we'll make this cloth by going to the Object context (F7) → Physics sub-context. Scroll down until you see the Cloth panel, and press the Cloth button. Now, a lot of settings will appear, most of which we'll ignore for now.

That's all you need to do to set your cloth up for animating, but if you hit AltA, your lovely fabric will just drop very un-spectacularly. That's what we'll cover in the next two sections about pinning and colliding.

Using Simulation to Shape/Sculpt a Mesh

You can Apply the Cloth modifier at any point to freeze the mesh in position at that frame. You can then re-enable the cloth, setting the start and end frames from which to run the simulation forward.

Another example of aging is a flag. Define the flag as a simple grid shape and pin the edge against the flagpole. Simulate for 50 frames or so, and the flag will drop to its "rest" position. Apply the Cloth modifier. If you want the flag to flap or otherwise move in the scene, re-enable it for the frame range when it is in camera view.

Smoothing of Cloth

Now, if you followed this from the previous section, your cloth is probably looking a little blocky. In order to make it look nice and smooth like the picture you need to apply a Smooth and/or Subsurf modifier in the Modifiers panel under the Editing context (F9). Then, in the same context, find the Links and Materials panel (the same one you used for vertex groups) and press Set Smooth.

Now, if you hit AltA, things are starting to look pretty nice, don't you think?

Cloth on armature

Cloth deformed by armature and also respecting an additional collision object: Regression blend file.

Cloth with animated vertex groups

Cloth with animated pinned vertices: Regression blend file. UNSUPPORTED: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will not work (e.g. from goal = 0 to goal = 0.5).

Cloth with Dynamic Paint

Cloth with Dynamic Paint using animated vertex groups: Regression blend file. UNSUPPORTED: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will not work (e.g. from goal = 0 to goal = 0.5) because the necessary "goal springs" cannot be generated on the fly.

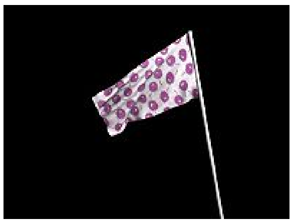
Using Cloth for Softbodies

Using cloth for softbodies.

Cloth can also be used to simulate softbodies. It's for sure not its main purpose but it works nonetheless. The example image uses standard Rubber material, no fancy settings, just AltA.

Blend file for the example image: Using Cloth for softbodies.

Cloth with Wind



Flag with wind applied.

Regression blend file for Cloth with wind and self collisions (also the blend for the image above): Cloth flag with wind and selfcollisions.

Collisions

Particles, Soft Bodies and Cloth objects may collide with mesh objects. Boids try to avoid Collision objects.

- The objects need to share at least one common layer to have effect.
- You may limit the effect on particles to a group of objects (in the Field Weights panel).
- *Deflection* for softbody objects is difficult, they often penetrate the colliding objects.
- Hair particles ignore deflecting objects (but you can animate them as softbodies which take deflection into account).

If you change the deflection settings for an object you have to recalculate the particle, softbody or cloth system (Free Cache), this is not done automatically. You can clear the cache for all selected objects with CtrlB → Free cache selected.

Mode: Object Mode

Panel: Object context → Physics sub-context → Collision

Options

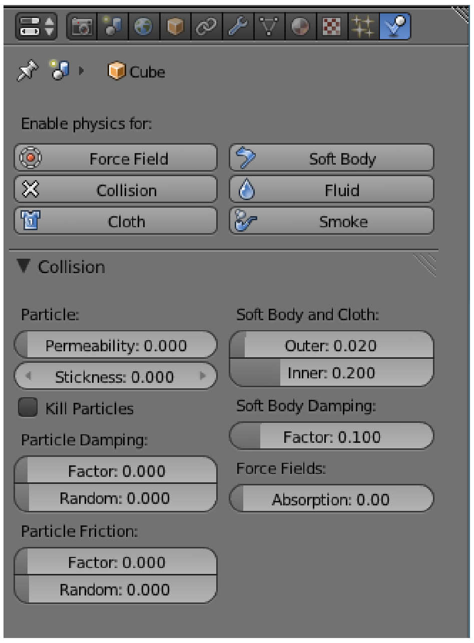


Image 1: Collision panel in the Physics sub-context.

Permeability
: Fraction of particles passing through the mesh. Can be animated with Object Ipos, Perm channel.

Stickiness
: How much particles stick to the object.

Kill Particles
: Deletes Particles upon impact.

Damping Factor
: Damping during a collision (independent of the velocity of the particles).

Random damping
: Random variation of damping.

Friction Factor
: Friction during movements along the surface.

Random friction
: Random variation of friction.

Image 1b: A softbody vertex colliding with a plane.

Soft Body and Cloth Interaction

Outer
: Size of the outer collision zone.

Inner
: Size of the inner collision zone (padding distance). **Only effective for softbodies**, cloth only use outer collision value.

Outside and inside is defined by the face normal, depicted as blue arrow in (*Image 1b*).

Damping Factor
: Damping during a collision.

Softbody collisions are difficult to get perfect. If one of the objects move too fast, the soft body will penetrate the mesh. See also the section about Soft Bodies.

Force Field Interaction

Absorption
: A deflector can also deflect effectors. You can specify some collision/deflector objects which deflect a specific portion of the effector force using the Absorption value. 100% absorption results in no force getting through the collision/deflector object at all. If you have 3 collision object behind each other with e.g. 10%, 43% and 3%, the absorption ends up at around 50% (`100×(1-0.1)×(1-0.43)×(1-0.03)`).

Examples

Image 2: Deflected Particles.

Here is a Meta object, dupliverted to a particle system emitting downwards, and deflected by a mesh cube:

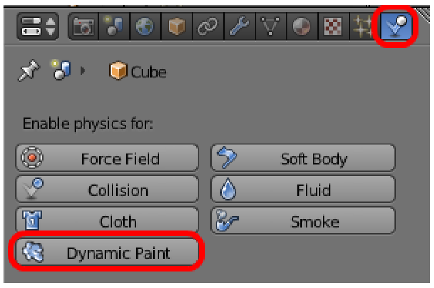
Hints

- Make sure that the normals of the mesh surface are facing towards the particles/points for correct deflection.
- Hair particles react directly to force fields, so if you use a force field with a short range you don’t need necessarily collision.
- Hair particles avoid their emitting mesh if you edit them in Particle mode. So you can at least model the hair with collision.

Dynamic Paint

Dynamic paint is a new modifier and physics system that can turn objects into paint canvases and brushes, creating vertex colors, image sequences or displacement. This makes many effects possible that were previously difficult to achieve, for example footsteps in the snow, raindrops that make the ground wet, paint that sticks to walls, or objects that gradually freeze.

This guide explains the very basics of Dynamic Paint user interface and general features.



How to activate the Dynamic Paint

Activating the modifier

Dynamic Paint can be activated from the "Physics" tab of the "Properties" editor.

Types

Modifier itself has two different types:

Canvas
: Makes object receive paint from Dynamic Paint brushes.

Brush
: Makes object apply paint on the canvas.

Note
You can also enable brush and canvas simultaneously. In that case same object's "brush" doesn't influence it's "canvas", but can still interact with other objects in the scene.

See also

- A step-by step introduction
- A detailed guide that covers every setting with images and examples (Currently not up-to-date)

Explode Modifier

Mode: Any Mode

Panel: Editing Context → Modifiers

Hotkey: None

The Explode Modifier is used to alter the mesh geometry (by moving/rotating its faces) in a way that (roughly) tracks underlying emitted particles that makes it look as if the mesh is being exploded (broken apart and pushed outward).

For the Explode Modifier to have a visible effect on the underlying mesh it has to be applied to a mesh which has a particle system on it, in other words it has to be a mesh which outputs particles. This is because the particle system on the mesh is what controls how a mesh will be exploded, and therefore without the particle system the mesh wont appear to alter. Also both the number of emitted particles and number of faces determine how granular the Explode Modifier will be. With default settings the more faces and particles the more detailed the mesh exploding will be, because there are more faces and particles to affect detachment/movement of faces.

Here is a link to an Ogg Theora Movie showing a cube with a particle system and Explode Modifier applied:

Media:Manual - Explode Modifier - Exploding Cube - 2.5.ogg

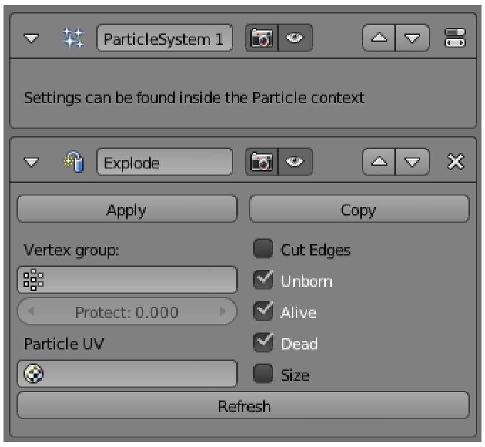
Here is a link to the original Blender file which has an Exploding cube setup, just free the particle cache by pressing the Free Bake button in the bake panel and then press the Animate button to see the animation:

Media:Manual_-_Explode_Modifier_-_Exploding_Cube_-_2.5.blend

Stacking Order Importance

This modifier is highly affected by its position within the modifier stacking order. If it is applied before a Particle System modifier it will not be affected by particles and therefore appear to do nothing. The Particle System Modifier must appear before the Explode Modifier because the Particle System Modifier has the information needed to drive the Explode Modifier.

Options



Settings can be found inside the Particle context

Vertex group:
Protect: 0.000
Particle UV
Cut Edges [ ]
Unborn [✓]
Alive [✓]
Dead [✓]
Size [ ]

Explode Modifier panel with Particle System Modifier above it

Vertex group
: If a mesh that has an Explode Modifier on it also has vertex groups assigned to it then this field will allow the selection of one of those vertex groups. This will indicate to the Explode Modifier that it should take into account the weight values assigned to areas of the selected vertex group.

Protect
: Clean vertex group edges. Depending on the weights assigned to that vertex group; either completely protect those faces from being affected by the Explode Modifier (which would happen if the faces had a weight value of 1) or completely remove protection from those faces (which would happen if the faces had a weight value 0).

Particle UV
: UV map to change with particle age.

Cut Edges
: Cut face edges for nicer shrapnel

Unborn
: Show mesh when particles are unborn

Alive
: Show mesh when particles are alive

Dead
: Show mesh when particles are dead

Size
: Use particle size for shrapnel

Refresh

Refresh data in the explode modifier

Fluid Simulation

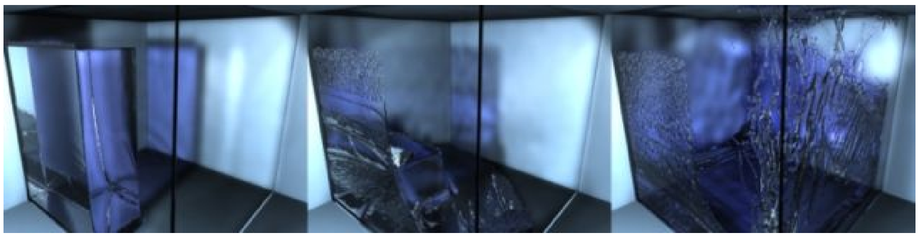
Mode: Object mode / Edit mode (Mesh)

Panel: Physics sub-context → Fluid

Description

While modeling a scene with blender, certain objects can be marked to participate in the fluid simulation, e.g. as fluid or as an obstacle. The bounding box of another object will be used to define a box-shaped region to simulate the fluid in (the so called "simulation domain"). The global simulation parameters (such as viscosity and gravity) can be set for this domain object.

Using the BAKE button, the geometry and settings are exported to the simulator and the fluid simulation is performed, generating a surface mesh together with a preview for each animation frame, and saving them to hard disk. Then the appropriate fluid surface for the current frame is loaded from disk and displayed or rendered.


A breaking dam.

Workflow

In general, you follow these steps:

- set the simulation domain (the portion of the scene where the fluid will flow),
- set the fluid source(s), and specify its material, viscosity, and initial velocity,
- eventually, set other objects to control the volume of the fluid (inlets and outlets),
- eventually, set other objects related to the fluid, like:
  - obstacles,
  - particles floating on the fluid,
  - fluid control, to shape part of the fluid in the desired form,
- eventually, animate the fluid properties,
- Bake the simulation (eventually, revise as necessary and bake repeatedly).

Baking is done on the Domain object!

When you calculate the fluid simulation, **you bake the simulation on the domain object**.

For this reason:

- all the baking options are visible only when selecting the Domain Object,
- baking options are explained in the the baking section of the Domain manual page.

More about the simulation

To know more about simulating fluids in Blender you can read:

- some useful hint about the simulation,
- some technical details, to learn how to do a more realistic fluid simulation,
- the fluids appendix to learn limitations and workarounds, and some additional links.

Ocean Simulation

Blender's ocean simulation tools take the form of a modifier, to simulate and generate a deforming ocean surface, and associated texture, used to render the simulation data. Ported from the open source Houdini Ocean Toolkit, it is intended to simulate deep ocean waves and foam.

Simulation Internals

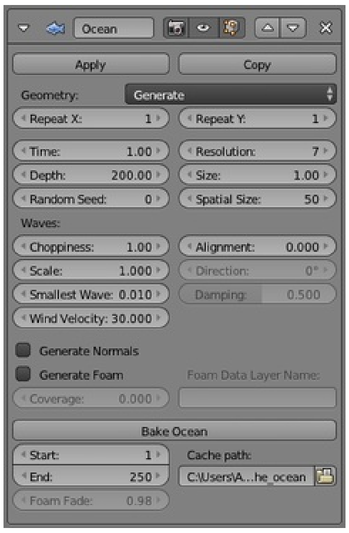
The simulator itself uses FFT methods to generate 2d grids of sim information internally, very similar to 2d texture maps. The simulator can generate three types of data - displacement, normals, and extra data that is used to calculate wave crest intersections (i.e. foam). After simulation, these maps are used to displace the ocean surface geometry in 3d, and also can be used for shading via the Ocean texture. The internal simulation engine is multithreaded with OpenMP to take advantage of multiple cores.

Ocean Modifier

Mode: Object mode

Panel: Modifiers context

Description


Ocean

Apply | Copy

Geometry: Generate

Repeat X: 1 | Repeat Y: 1

Time: 1.00 | Resolution: 7

Depth: 200.00 | Size: 1.00

Random Seed: 0 | Spatial Size: 50

Waves:

Choppiness: 1.00 | Alignment: 0.000

Scale: 1.000 | Direction: 0°

Smallest Wave: 0.010 | Damping: 0.500

Wind Velocity: 30.000

Generate Normals

Generate Foam | Foam Data Layer Name:

Coverage: 0.000

Bake Ocean

Start: 1 | Cache path:

End: 250 | C:\Users\A...he_ocean

Foam Fade: 0.98

Ocean Modifier Panel
The Ocean Modifier is the main place in Blender where the simulation is performed - the modifier stores the simulation data, and applies it to create a deformed ocean surface mesh. The ocean modifier can also add a vertex color channel, in order to visualize a foam map.

Geometry Options

Geometry
: The ocean modifier can affect mesh geometry by:

- Generating a tiled mesh grid that exactly corresponds with the resolution of the sim data

When generating a mesh surface, the existing mesh object is completely overridden with the ocean grid. A UV channel is also added, mapping the [0.0,1.0] UV space to the simulation grid.

- Displacing an existing mesh of arbitrary topology

Repeat X, Repeat Y
: When generating a mesh surface, controls the number of times the grid is tiled in X and Y directions. UVs for these tiled mesh areas continue outside of the [0,1] UV space.

Simulator Options

Time
: The time at which the ocean surface is being evaluated. To make an animated ocean, you will need to insert keyframes (RMB) and animate this time value - the speed that the time value is changing will determine the speed of the wave animation

Resolution
: The main control of quality vs speed in the simulation engine, this determines the resolution of the internal 2D grids generated by the sim. The internal grids are powers of two of the resolution value, so a resolution value of 16 will create simulation data of size 256x256. The higher the resolution, the slower it will be to calculate, but the more detail will be available to use.

: Note: When using the 'Generate' modifier geometry option, this resolution value also determines the resolution of the generated mesh surface, equal to the resolution of the internal sim data.

Spatial Size
: The width of the ocean surface area being simulated, in meters. This also determines the size of the generated mesh, or the

displaced area, in Blender units. Of course you can scale the object with ocean modifier in object mode to tweak the apparent size in your scene.

Depth
: The constant depth of the ocean floor under the simulated area

Wave Options

Choppiness
: The choppiness of the wave peaks. With a choppiness of 0, the ocean surface is only displaced up and down in the Z direction, but with higher choppiness, the waves are also displaced laterally in X and Y, to create sharper wave peaks.

Scale
: An overall scale control for the amplitude of the waves. It approximates the height or depth of the waves above or below zero. Rather than just scaling the ocean object in Z, it scales all aspects of the simulation, displacement in X and Y, and corresponding foam and normals too.

Alignment
: The directionality of the wave shapes due to wind. At a value of 0, the wind and waves are randomly, uniformly oriented. With higher Alignment values, the wind is blowing in a more constant direction, making the waves appear more compressed and aligned to a single direction.

Direction
: When using Alignment, the direction in degrees that the waves are aligned to.

Damping
: When using Alignment, amount that inter-reflected waves are damped out. This has the effect of making the wave motion more directional (not just the wave shape). With damping of 0.0, waves are reflected off each other every direction, with damping of 1.0, these inter-reflected waves are damped out, leaving only waves traveling in the direction of the wind.

Smallest Wave
: A minimum limit for the size of generated waves. Acts similarly to a low-pass filter, removing higher frequency wave detail.

Wind Velocity
: Wind speed in meters/second. With a low velocity, waves are restricted to smaller surface waves.

Sim Data Generation Options

Using foam vertex colors with a named data layer

By default, the simulator only generates displacement data, since it takes the least amount of work and gives the fastest feedback. Additional sim data can be generated for rendering as well.

Generate Normals
: Simulates additional normal map data. This can be used by the Ocean texture, when mapped to Normals, as a bump map, and enables generating normal map image sequences when baking.

Generate Foam
: Simulates additional foam data. This can be retrieved by the Ocean texture for use in texturing (perhaps as a mask), and enables generating foam map image sequences when baking.

Coverage
: Tweaks the amount of foam covering the waves, negative values will reduce the amount of foam (leaving only the topmost peaks), positive values will add it.

Foam Data Layer Name
: Optional name for the vertex data layer, used by the Ocean modifier to store foam maps as vertex colors. This is required for accessing the foam data in the renderer.

Baking

Rather than simulating the ocean data live, the ocean data can be baked to disk. When a simulation is baked, the simulator engine is completely bypassed, and the modifier/texture retrieves all information from the baked files.

Baking can be advantageous for a few reasons:

- It's faster to use the stored data rather than re-calculating it
- Allows rendering ocean data in external renderers

- Enables more advanced foam maps

Data Files

Sim data is stored in disk as sequences of OpenEXR image maps, one for each of displacement, normal and foam (if enabled to be generated). Upon loading the data from these baked files, when a frame of the bake sequence is read from disk, it is cached in memory. This means that accessing loaded frames subsequent times is fast, not incurring the overhead of disk access.

Since these baked files are plain OpenEXRs, they can also be opened and rendered in any other application or renderer that supports them.

Baking Foam

Baking also provides improved foam capabilities. When simulating live, the ocean simulator retrieves data for that current frame only. In the case of the foam map, this represents the tips of wave crests for that given frame. In reality, after foam is created by wave interactions, it remains sitting on the top of the wave surface for a while, as it dissipates. With baking, it's possible to approximate that behaviour, by accumulating foam from previous frames, leaving it remaining on the surface.

[video link]

Baking Options

Start, End
: Frames of the simulation to bake (inclusive). The start and end frames of the bake are repeated when accessing frames outside the baked range.

Cache Path
: Folder to store the baked EXR files in. The sequences will be in the form disp_####.exr, normal_####.exr, and foam_####.exr where #### is the four digit frame number. If the cache path folder does not exist, it will be created.

[video link]

Simulated and baked to image maps in Blender, rendered in 3Delight.

History

The core simulator was developed by Drew Whitehouse, for the Houdini Ocean Toolkit. This was ported to C by Hamed Zaghaghi and integrated in a patch for the Blender 2.4 series, sponsored by ProMotion Studios/Red Cartel during production of the short film Lighthouse.

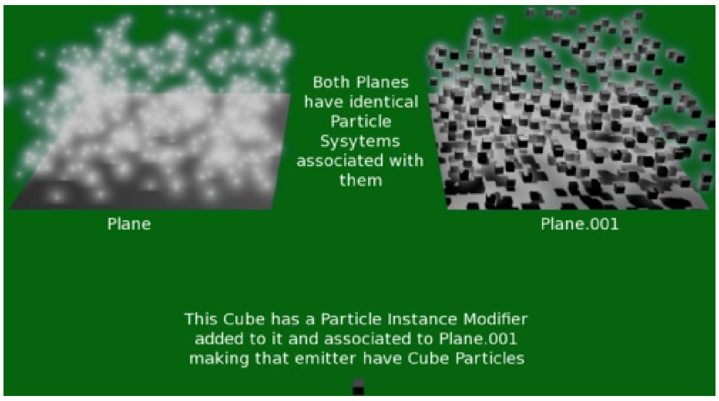
In this work, Matt Ebb re-integrated the core simulator for Blender 2.5, and added additional functionality, fixes, and optimisations, sponsored by the 'Save the Ocean Sim' project.

Particle Instance Modifier

Mode: Any mode

Panel: Modifiers (Editing context, F9)

When a ParticleInstance modifier is added to an object, that object will be used as a particle shape on an object which has a particle system associated with it. This means that to use this modifier you must also have another object which has a particle system on it, otherwise the ParticleInstance modifier will appear to do nothing.

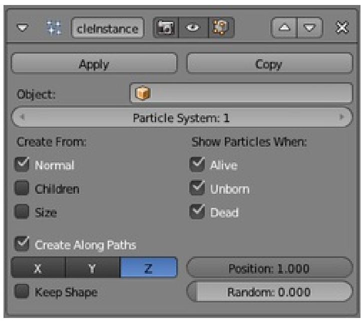

Both Planes have identical Particle Sysytems associated with them
Plane
Plane.001
This Cube has a Particle Instance Modifier added to it and associated to Plane.001 making that emitter have Cube Particles

Particle system on left has no ParticleInstance modified object associated with it. The one on the right is associated with cube shown by using a ParticleInstance modifier on the cube.

Here is a brief explanation of the various terms and definition used in relation to particles and the ParticleInstance modifier:

- Particle system – An object (mesh) which has the ability to emit/generate particles activated on it.
- Normal particle – A particle that is not a children/child generated particle type.
- Children/child particle – A particle type that is generated and placed with relation to other normal particles that already exist. Children particles are generally much quicker to calculate.
- Unborn particle – A particle which has not yet been displayed/emitted because it is not its time to be emitted/displayed. One of the reasons a particle can be in unborn state is that it is before the frame at which it is to be emitted.
- Alive particle – A particle which has been displayed/emitted and has not yet reached its dead state. One of the reasons a particle can be in an alive state is that it has been alive for less frames than its life length.
- Dead particle – A particle which has been displayed/emitted and has reached its end of life length and at that point it enters the dead state.

Options



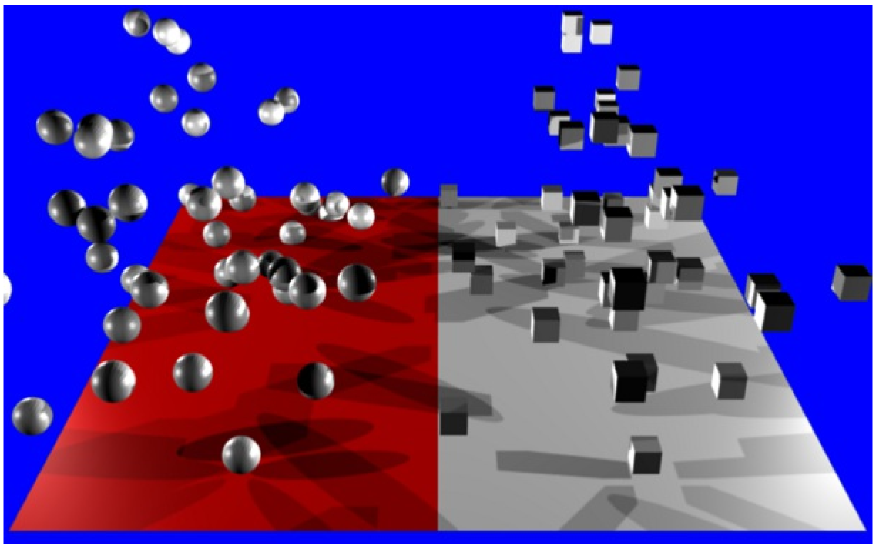
Particle Instance Modifier

Because of the co-dependant way in which the ParticleInstance modifier is influenced by the underlying particle systems on other objects, some of the apparent effects generated by the ParticleInstance modifier can look and act vastly different, depending on the underlying settings of the particle systems it is associated with. This is worth taking account of if the ParticleInstance modifier settings don’t appear to be giving the results expected, as it may indicate that the particle system settings may need altering rather than the ParticleInstance modifier settings.

Object
: The Object field, associates this ParticleInstance modifier with another object (usually an object having a particle system…). This indicates that when the object named in this field emits particles, those particles will have the mesh shape of the current ParticleInstance modifier’s mesh.
If for example a sphere has a ParticleInstance modifier added to it, when the Object field of this modifier is filled in with the name of an object that emits particles, those particle will be sphere shaped.
Even though most of the time the Object field will have the name of an object with a particle system, this is not mandatory, you can enter an object’s name which does not have a particle system, and it will be accepted by the Object field, as there do not appear to be any checks made to make sure the object’s name entered into this field is “valid”.

Particle System
: The Particle System field is used to select which particle system number to apply the ParticleInstance modifier to, when the

mesh which has the particle system on it has more than one of these. The Particle System field can have a value between **1** and **10**. It is possible to select any of the ten particle system numbers, however a check will **not** be made with the underlying particle emitting object specified previously in the Object field. If you select a particle system number which does not exist on the particle emitting object, then the particles on the emitting mesh will keep their normal particle shapes – no warning will be given that the chosen particle system does not exist on a particular particle emitting mesh.
As an example, below is a single plane mesh with two areas (the first area shown in red and the second in white), with different particle systems applied to each area. The left side using a ParticleInstance modifier which has the shape of a sphere and the right side having a ParticleInstance modifier which has the shape of a cube.


Render showing a single Plain mesh object assigned to two different vertex groups and each of those vertex groups is assigned a separate and independent particle system, with each particle system being assigned a different ParticleInstance modifier. In the case shown the ParticleInstance modifiers are a sphere and a cube.
Example Blend file

Creation

Normal
: When selected, the Normal button tells the ParticleInstance modifier to draw instances of itself wherever normal particle types are emitted from the underlying particle system. So if the current ParticleInstance modifier is a sphere shape, when normal particles are emitted they will be spheres.

Children
: When selected, the Children button tells the ParticleInstance modifier to draw instances of itself wherever children/child particles are emitted/used on the underlying particle system. So if the current ParticleInstance modifier is a sphere shape, when children/child particles are emitted they will be spheres.

Size
: Scale the instanced objects by the particle size attribute. When this is disabled, all the copies appear the same size as the origin.

Display

Unborn
: When selected, the Unborn button tells the ParticleInstance modifier to draw instances of itself wherever unborn particles will be emitted/used on the underlying particle system. So if the current ParticleInstance modifier is a sphere shape, when unborn particles are present they will be spheres.

Alive
: When selected, the Alive button tells the ParticleInstance modifier to draw instances of itself wherever alive particles will be emitted/used on the underlying particle system. So if the current ParticleInstance modifier is a sphere shape, when alive particles are present they will be spheres.

Dead
: When selected, the Dead button tells the ParticleInstance modifier to draw instances of itself wherever dead particles will occur on the underlying particle system. So if the current ParticleInstance modifier is a sphere shape, when dead particles are present they will be spheres.

Using Paths

Create Along Paths
: This option tries to make the underlying mesh object of the Particle Instance modifier deform its mesh shape in such a way as to try and match the path traveled by the particles/hair strands of the system associated with it.
For example, below is a screen shot showing the path of a single keyed particle as it travels its way through each of the different way points **1** to **4** (target particle systems), when it reaches way point **4** the particle dies and ends its journey.

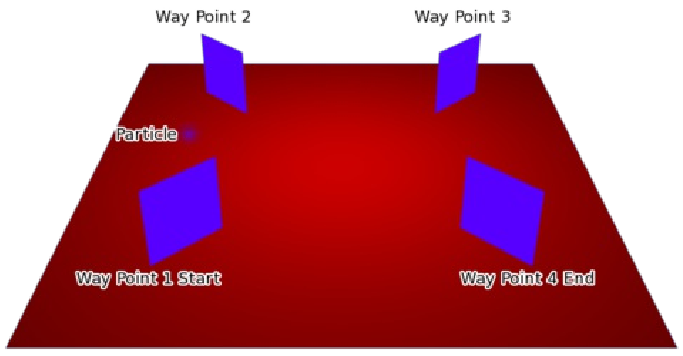
X,Y,X Rotation Axis

Specify which pole axis to use for the rotation.

Keep Shape
: Enabling this prevents the object from being deformed. It instead simply aligns to the end of the path at the object's center.

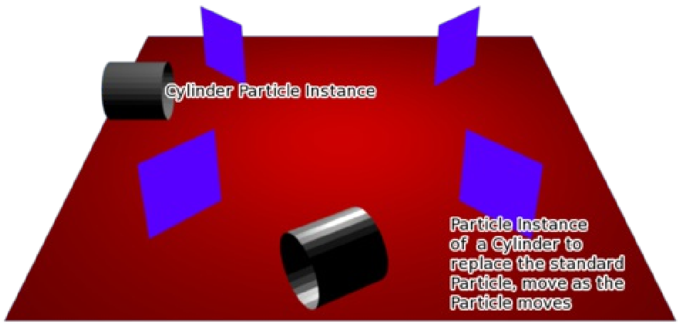
Position
: Specify what percentage of the path the object fills. You could create a growing effect by animating this value over time.

Random
: Scales the position value of each instance a random value.



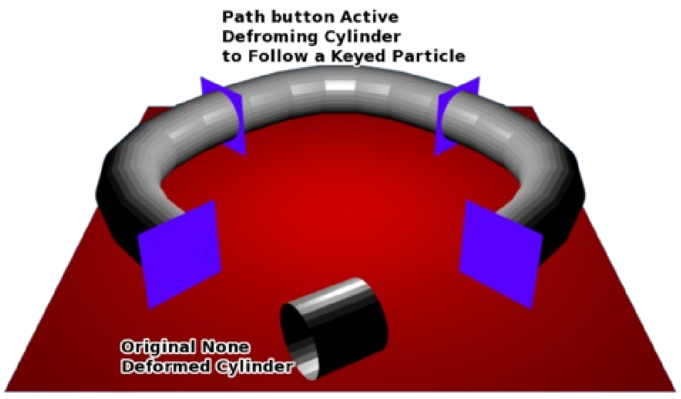
Keyed particle following way points (showing one particle).
Example Blend file

When a ParticleInstance modifier is added to a cylinder object and then associated with the way point particle system, the particle position is copied by the cylinder and placed at the particles position. So the mesh object follows the location of the particle. The cylinder does not alter any of its other properties when following the particle, only the cylinders location gets altered, shape and rotation do not get altered. See screenshot below:


Cylinder Particle Instance
Particle Instance of a Cylinder to replace the standard Particle, move as the Particle moves

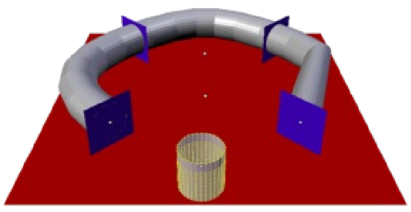
Keyed particle following way points showing a mesh object (ParticleInstance modifier) in place of the original particle.
Example Blend file

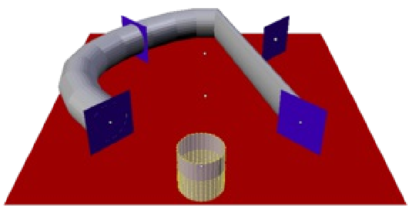
Both of the above examples had the ParticleInstance modifier Path button deactivated.
When the Path button is activated the effect can be seen in the screenshot below:

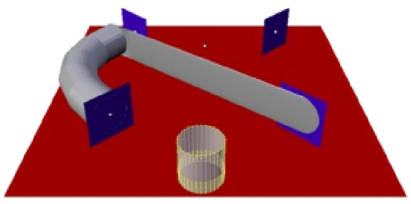


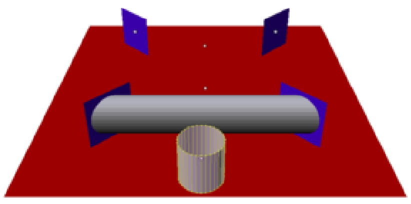
Keyed particle following way points showing a mesh object (ParticleInstance modifier) in place of the original particle, that is also being deformed to fit the travel path of the original particle.
Example Blend file

Instead of the cylinder location just following the position of the particle (and not altering its shape), the cylinder tries to fit its mesh to the shape of the path followed by the particle.
The mesh geometry of the object which is trying to deform can have an impact on how well the deformation is carried out. In the case of the cylinder, it has many loop cuts along its length so that it can bend at those points to deform along the particle path.
For example here is the same scene with the number of loop cuts along the length of the cylinder reduced, showing the effect on the deformation of the cylinder along the particle path.

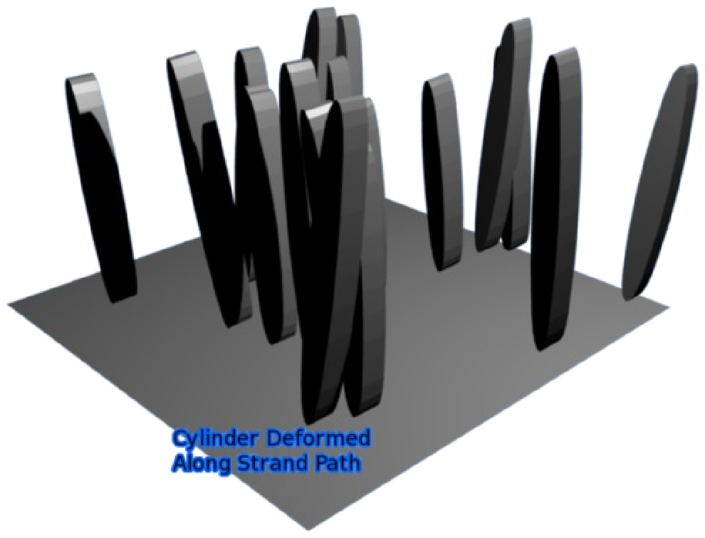

The cylinder has most of its edge loops so most of the path deform is very regular apart from at the very end of the curve.


The cylinder has some of its edge loops removed so the path of the deform starts to become less regular.


Now the deform path is very rough.


At this point there aren’t any vertices to bend the cylinder to follow the path, and instead the cylinder just goes directly to the last way point **4**.

Once all the extra edge loops around cylinder are removed so that there is only the top and bottom vertices left, meaning that the cylinder doesn’t have enough geometry to bend, in that case it cannot follow the path of the particle, so it just goes from the start way point **1** to the ending way point **4**.
The ParticleInstance modifier Path button works for hair (strand) particles as well as with keyed particles. In this case the mesh of the ParticleInstance modifier will follow the length and profile of the hair strands paths.
Below is a screenshot showing the effect of the Path button on hair:



Strand with a ParticleInstance modifier associated with it and deforming the cylinder along the hair profile.
Example Blend file

Note
Strands when they are generated instantly die when created so for the Path button to be of any use, you must also have the Dead button activated. Otherwise the path a mesh took will not be visible!

See Also

- Particles

Particles

Particles are lots of items emitted from mesh objects, typically in the thousands. Each particle can be a point of light or a mesh, and be joined or dynamic. They may react to many different influences and forces, and have the notion of a lifespan. Dynamic particles can represent fire, smoke, mist, and other things such as dust or magic spells.

Hair type particles are a subset of regular particles. Hair systems form strands that can represent hair, fur, grass and bristles.

You see particles as a Particle modifier, but all settings are done in the Particle tab.

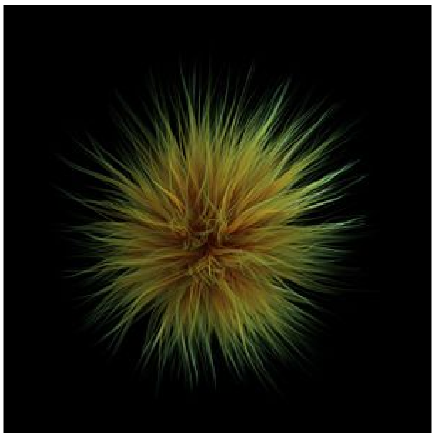


Image 1: Some fur made from particles (Blend file).

Particles generally flow out from their mesh into space. Their movement can be affected by many things, including:

- Initial velocity out from the mesh.
- Movement of the emitter (vertex, face or object) itself.
- Movement according to “gravity” or “air resistance”.
- Influence of force fields like wind, vortexes or guided along a curve.
- Interaction with other objects like collisions.
- Partially intelligent members of a flock (herd, school, …), that react to other members of their flock, while trying to reach a target or avoid predators.
- Smooth motion with softbody physics (only Hair particle systems).
- Or even manual transformation with Lattices.

Particles may be rendered as:

- Halos (for Flames, Smoke, Clouds).
- Meshes which in turn may be animated (e.g. fish, bees, …). In these cases, each particle “carries” another object.
- Strands (for Hair, Fur, Grass); the complete way of a particle will be shown as a strand. These strands can be manipulated in the 3D window (combing, adding, cutting, moving, etc).

Every object may carry many particle systems. Each particle system may contain up to 100.000 particles. Certain particle types (Hair and Keyed) may have up to 10.000 children for each particle (children move and emit more or less like their respective parents). The size of your memory and your patience are your practical boundaries.

Incompatibility with Prior Versions
There are many differences between the “old” particle system that was used up to and including version 2.45, and the “new” particle system. There are many things possible now that could not be done with the old system. The new system is incompatible to the old system, though Blender tries to convert old particle systems, which works only to some extent. The old system is most like the new Emitter system (keep reading to find out what that is). If you are using an old version of Blender 2.45 and previous, click here to access the old documentation.

Workflow

The process for working with standard particles is:

1. Create the mesh which will emit the particles.
2. Create one or more Particle Systems to emit from the mesh. Many times, multiple particle systems interact or merge with each other to achieve the overall desired effect.
3. Tailor each Particle System’s settings to achieve the desired effect.
4. Animate the base mesh and other particle meshes involved in the scene.
5. Define and shape the path and flow of the particles.
6. For Hair particle systems: Sculpt the emitter’s flow (cut the hair to length and comb it for example).
7. Make final render and do physics simulation(s), and tweak as needed.

Creating a Particle System

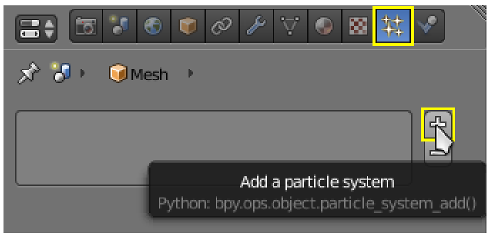


Image 2: Adding a particle system.

To add a new particle system to an object, go to the Particles tab of the object Settings editor and click the small + button. An object can have many Particle Systems.

Each particle system has separate settings attached to it. These settings can be shared among different particle systems, so one doesn't have to copy every setting manually and can use the same effect on multiple objects. Using the Random property they can be randomized to look slightly different, even when using the same settings.

Types of Particle systems

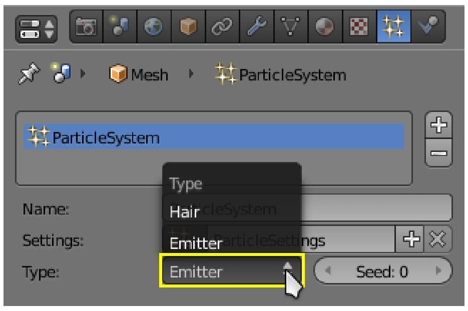


Image 3: Particle system types.

After you have created a particle system, the Property window fills with many panels and buttons. But don’t panic! There are two different types of particle systems, and you can change between these two with the Type drop-down list:

Emitter
: This parallels the old system to the greatest extent. In such a system, particles are emitted from the selected object from the Start frame to the End frame and have a certain lifespan.

Hair
: This system type is rendered as strands and has some very special properties: it may be edited in the 3D window in realtime and you can also animate the strands with Cloth Simulation.

The settings in the Particle System panel are partially different for each system type. For example, in *Image 3* they are shown for only system type Emitter.

Common Options

Each system has the same basic sets of controls, but options within those sets vary based on the system employed. These sets of controls are:

Emission	Settings for the initial distribution of particles on the emitter and the way they are born into the scene.
Cache	In order to increase realtime response and avoid unnecessary recalculation of particles, the particle data can be cached in memory or stored on disk.
Velocity	Initial speed of particles.
Rotation	Rotational behavior of particles.
Physics	How the movement of the particles behaves.
Render	Rendering options.
Display	Realtime display in the 3D View.
Children	Control the creation of additional child particles.
Field Weights	Factors for external forces.
Force Field Settings	Makes particles force fields.
Vertex Groups	Influencing various settings with vertex groups.

Links

- Tutorials
- Physics Caching and Baking
- Particle Rewrite Documentation
- Thoughts about the particle rewrite code
- Static Particle Fur Library

Smoke Simulation

Development notes

Blender's new smoke simulation is based on the paper 'Wavelet Turbulence for Fluid Simulation' and associated sample code.

It has been implemented in Blender by Daniel Genrich and Miika Hamalainen.

Inner working

The simulator uses a volumetric fluid-based model, with the end results output as voxel grids. This voxel data is visualized interactively in Blender's 3D view using custom OpenGL shading, and can be rendered using the Voxel Data texture. Blender's **smoke simulation** wraps Voxels around existing Particles. It requires a particle-emitting object and a 'domain' object within which smoke is rendered.

Note
This Part of the Documentation uses the 2.58 Release

User workflow

The smoke simulation is similar to the Fluid simulation: a Domain and Flow object is required to do a smoke simulation:

- set as the simulation Domain an object that defines the bounds of the simulation volume,
- set as the Flow object an object which determines where the smoke will be produced from,
- set Collision objects, to make the smoke interact with objects in the scene.
- assign a Material to the smoke
- save the project
- bake the simulation

In case you are having troubles, please consult the Appendix

Soft Bodies

Image 1a: A softbody cloth uncovering a text. Animation – Blend file

A Soft Body in general, is a simulation of a soft or rigid deformable object. In Blender, this system is best for simple cloth objects and closed meshes. There is dedicated Cloth Simulation physics that use a different solver, and is better for cloth.

This simulation is done by applying forces to the vertices or controlpoints of the object. There are exterior forces like gravity or forcefields and interior forces that hold the vertices together. This way you can simulate the shapes that an object would take on in reality if it had volume, was filled with something, and was acted on by real forces.

Soft Bodies can interact with other objects (*Collision*). They can interact with themselves (*Self Collision*).

The result of the Soft Body simulation can be converted to a static object. You can also *bake edit* the simulation, i.e. edit intermediate results and run the simulation from there.

Typical scenarios for using Soft Bodies

Image 1b: A wind cone. The cone is a Soft Body, as the suspension. Animation – Blend file

Soft Bodies are well suited for:

- Elastic objects with or without collision.
- Flags, fabric reacting to forces.
- Certain modeling tasks, like a cushion or a table cloth over an object.
- Blender has another simulation system for clothing (see Clothes). But you can sometimes use Soft Bodies for certain parts of clothing, like wide sleeves.
- Hair (as long as you minimize collision).
- Animation of swinging ropes, chains and the like.

The following videos may give you some more ideas: [1], [2]

Creating a Soft Body

Soft Body simulation works for all objects that have vertices or control points:

- Meshes.
- Curves.
- Surfaces.
- Lattices.

To activate the Soft Body simulation for an object:

- In the Properties window, go to the Physics tab (it is all the way on the right, and looks like a bouncing ball).
- Activate the Soft Body button.

A lot of options appear. For a reference of all the settings see this page.

- You start a Soft Body simulation with AltA.
- You pause the simulation with Space, continue with AltA.

- You stop the simulation with Esc.

Simulation Quality

The settings in the Soft Body Solver panel determine the accuracy of the simulation.

Min Step
: Minimum simulation steps per frame. Increase this value, if the Soft Body misses fast moving collision objects.

Max Step
: Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the Error Limit) but you have probably a good reason to change it.

Auto-Step
: Use Velocities for automatic step sizes.

Error Limit
: Rules the overall quality of the solution delivered. Default 0.1. The most critical setting that says how precise the solver should check for collisions. Start with a value that is 1/2 the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how “bad” it is doing and the Error Limit causes the solver to do some “adaptive step sizing”.

Fuzzy
: Simulation is faster, but less accurate.

Choke
: Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Diagnostics

Print Performance to Console
: Prints on the console how the solver is doing.

Estimate Matrix
: Estimate matrix. Split to COM , ROT ,SCALE

Cache and Bake

Soft Bodies and other physic simulations use a unified system for caching and baking. See Particle Cache for reference.

The results of the simulation are automatically cached to disk when the animation is played, so that the next time it runs, it can play again quickly by reading in the results from the disk. If you Bake the simulation the cache is protected and you will be asked when you’re trying to change a setting that will make a recalculating necessary.

Beware of the Start and End settings

The simulation is only calculated for the frames in-between the Start and End frames (Bake panel), even if you don’t actually bake the simulation! So if you want a simulation longer than the default setting of 250 frames you have the change the End frame.

- Caching:
  - As animation is played, each physics system writes each frame to disk, between the simulation start and end frames. These files are stored in folders with prefix “`blendcache`”, next to the .blend file.
  - The cache is cleared automatically on changes - but not on all changes, so it may be necessary to free it manually, e.g. if you change a force field. Note that for the cache to fill up, one has to start playback before or on the frame that the simulation starts.
  - If you are not allowed to write to the required sub-directory caching will not take place.
  - The cache can be freed per physics system with a button in the panels, or with the CtrlB shortcut key to free it for all selected objects.
  - You may run into trouble if your .blend file path is very long and your operating system has a limit on the path length that is supported.
- Baking:
  - The system is protected against changes after baking.
  - The Bake result is cleared also with CtrlB for all selected objects or click on Free Bake for the current Soft Body system.
  - If the mesh changes the simulation is not calculated anew.

For renderfarms, it is best to bake all the physics systems, and then copy the blendcache to the renderfarm as well.

Interaction in real time

To work with a Soft Body simulation you will find it handy to use the Timeline window. You can change between frames and the simulation will always be shown in the actual state. The option Continue Physics in the Playback menu of the Timeline window lets you interact in real time with the simulation, e.g. by moving collision objects or shake a Soft Body object. And this is real fun!

Continue Physics does not work while playing the animation with AltA

Right. This works only if you start the animation with the Play button of the Timeline window.

You can than select the Soft Body object while running the simulation and Apply the modifier in the Modifiers panel of the Editing context. This makes the deformation permanent.

Tips

- Soft Bodies work especially well if the objects have an even vertex distribution. You need enough vertices for good collisions. You change the deformation (the stiffness) if you add more vertices in a certain region (see the animation of *Image 1b*).
- The calculation of collisions may take a long time. If something is not visible, why calculate it?
- To speed up the collision calculation it is often useful to collide with an additional, simpler, invisible, somewhat larger object (see the example to *Image 1a*).
- Use Soft Bodies only where it makes sense. If you try to cover a body mesh with a tight piece of cloth and animate solely with Soft Body, you will have no success. Self collision of Soft Body hair may be activated, but that is a path that you have to wander alone. We will deal with Collisions in detail later.
- Try and use a Lattice or a Curve Guide Soft Body instead of the object itself. This may be magnitudes faster.

Links

- Developer Notes
- Swinging of a chain
- Softbodies for Rigged Characters

Introduction

Lighting is a very important topic in rendering, standing equal to modeling, materials and textures. The most accurately modeled and textured scene will yield poor results without a proper lighting scheme, while a simple model can become very realistic if skillfully lit.

Viewing Restrictions

The color of an object and the lighting of your scene is affected by:

- Your ability to see different colors (partial color blindness is common).
- The medium in which you are viewing the image (e.g. an LCD panel versus printed glossy paper).
- The quality of the image (e.g. a JPEG at **0.4** compression versus **1.0**).
- The environment in which you are viewing the image (e.g. a CRT monitor with glare versus in a dark room, or in a sunshiny blue room).
- Your brain’s perception of the color and intensity relative to those objects around it and the world background color, which can be changed using color manipulation techniques using Blender Composite Nodes.

Global Influences

In Blender, the elements under your control which affect lighting are:

- The color of the world ambient light.
- The use of Ambient Occlusion as a way to cast that ambient light onto the object.
- The degree to which the ambient light colors the material of the object.
- The use of Indirect lighting, where the color of one object radiates onto another.
- The render engine used (Blender Internal versus Yafray).
- The lamps in your scene.

The physics of light bouncing around in the real world is simulated by Ambient Occlusion (a world setting), buffer shadows (which approximate shadows being cast by objects), ray tracing (which traces the path of photons from a light source). Also, within Blender you can use Indirect lighting. Ray tracing, ambient occlusion, and indirect lighting are computer-intensive processes. Blender can perform much faster rendering with its internal scan line renderer, which is a very good scan line renderer indeed. This kind of rendering engine is much faster since it does not try to simulate the real behavior of light, assuming many simplifying hypotheses.

Lighting Settings

Only after the above global influences have been considered, do you start adding light from lamps in your scene. The main things under your control are the:

- Type of light used (Sun, Spot, Lamp, Hemi, etc.).
- Color of the light.
- Position of the light and its direction.
- Settings for the light, including energy and falloff.

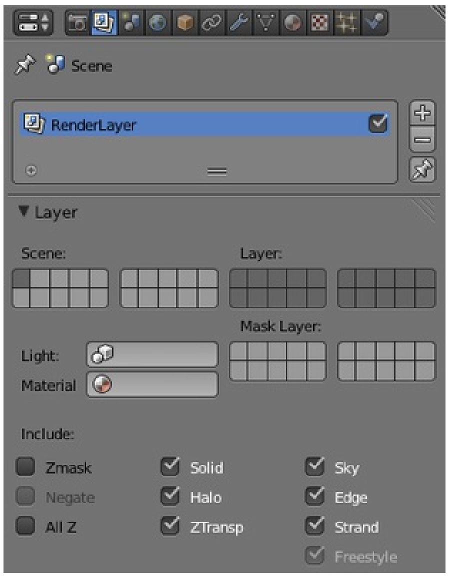
Then you are back to how that material’s shader reacts to the light.

This chapter attempts to address the above, including how lights can work together in rigs to light your scene. In this chapter we will analyze the different type of lights in Blender and their behavior; we will discuss their strong and weak points. We also describe many lighting rigs, including the ever-popular three-point light method.

Lighting in the Workflow

In this user manual we have placed Lighting before Materials; you should set up your lighting before assigning materials to your meshes. Since the material shaders react to light, without proper lighting, the material shaders will not look right, and you will end up fighting the shader, when it is really the bad lighting that is causing you grief. All of the example images in this section do not use any material setting at all on the ball, cube or background.

Overriding Materials to Reset Lighting



- [ ] Zmask
- [ ] Negate
- [ ] All Z
- [x] Solid
- [x] Halo
- [x] ZTransp
- [x] Sky
- [x] Edge
- [x] Strand
- [x] Freestyle

Material field in the Render Layers panel

If you have started down the road of assigning materials, and are now fiddling with the lighting, we suggest that you create a default, generic gray material--no Vertex Color, no Face Texture, no Shadeless, just plain old middle gray with RGB of (**0.8**, **0.8**, **0.8**). Name this “`Gray`”.

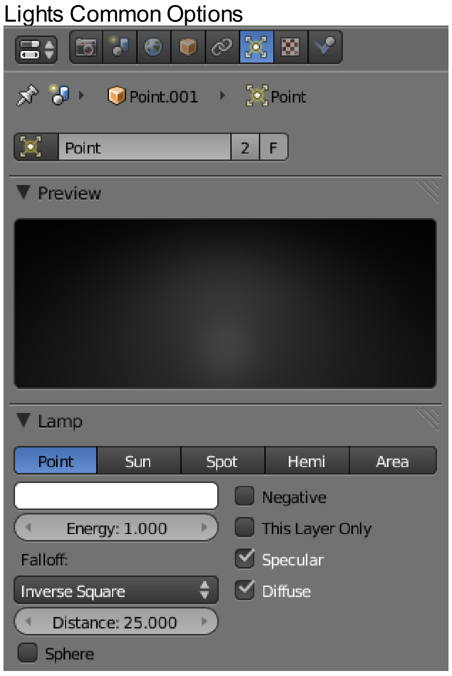
Next go to the Render context. In the Render Layers panel, select your new “`Gray`” material in the Material field. This will override any materials you may have set, and render everything with this color. Using this material, you can now go about adjusting the lighting. Just empty this field to get back to your original materials.

Lights

As previously stated, there are multiple types of lighting in Blender, like indirect light or ambient light. However, one of the most used are “lights”, or “lamps”. In this section, we will discuss general info and settings for these lights (you will find more lamp-specific details in the Lamps section):

- Light Properties – settings common to all lamps.
- Light Attenuation.
- Textures – how to apply texture(s) to lamps.
- What Light Affects.
- Lights In Other Contexts – lamp-related setting in other contexts.

Lights Common Options



Lamp Properties panels

There are five types of lamps in Blender. They share all or some of the options listed here:

Object Data

Browse Light Object Data
: Click to view all lights in the current scene.

Name
: The name of the currently selected light object data. Edit to change the name.

Number of Users
: The number of light objects sharing the light object data.

F
: Create a fake user for this object data.

Preview

A quick preview of the light settings.

Lamp

Distance
: The Dist field indicates the number of Blender Units (BU) at which the intensity of the current light source will be half of its intensity. Objects less than the number of BU away from the lamp will get more light, while objects further away will receive less light. Certain settings and lamp falloff types affect how the Distance field is interpreted, meaning that it will not always react the same; see the page about light falloff.

- The Sun and Hemi Lamps are another class of Lamps which uses a constant falloff. Those lamps don't have a Dist field, and are often called "Base Lighting Lamps".

Energy
: The intensity of the light source's illumination (from **0.0** to **10.0**).

Color
: The color of the light source's illumination. Opens a color swatch.

Negative
: Let the lamp cast negative light.

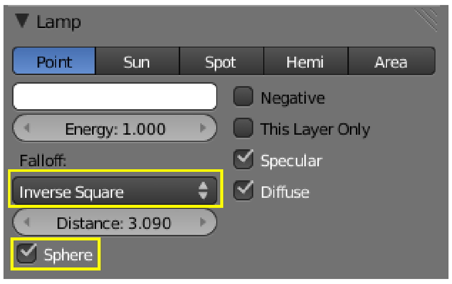
This Layer Only
: The Lamp only illuminates objects on the same layer the lamp is on.

Specular
: The Lamp creates specular highlights.

Diffuse
: The Lamp does diffuse shading.

Light Attenuation

Description



Negative
Energy: 1.000
This Layer Only
Falloff:
Specular ✓
Inverse Square
Diffuse ✓
Distance: 3.090
Sphere ✓

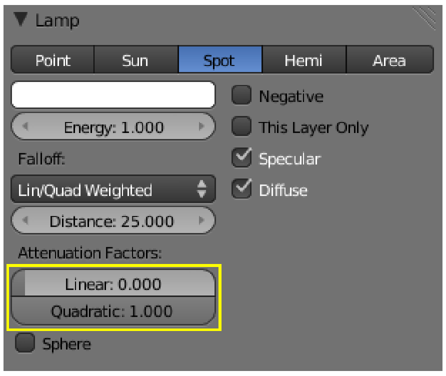
Lamp panel, falloff options highlighted

There are two main controls for light falloff for Point and Spot lamps.

- The lamp Falloff type drop-down list, and
- The Sphere button.

Falloff types

Lin/Quad Weighted



Negative
Energy: 1.000
This Layer Only
Falloff:
Specular ✓
Lin/Quad Weighted
Diffuse ✓
Distance: 25.000
Attenuation Factors:
Linear: 0.000
Quadratic: 1.000
Sphere

Lamp panel with Lin/Quad Weighted Falloff options highlighted

When this setting is chosen, two sliders are shown, Linear and Quadratic, which control respectively the “linearness” and “quadraticness” of the falloff curve.

This lamp falloff type is in effect allowing the mixing of the two light attenuation profiles (linear and quadratic attenuation types).

Linear

This slider input field can have a value between **0.0** and **1.0**. A value of **1.0** in the Linear field and **0.0** in the Quadratic field in effect means that the light from this source is completely linear. This means that at the number of Blender Units distance specified in the Distance field, this light source's intensity will be half the value it was originally.

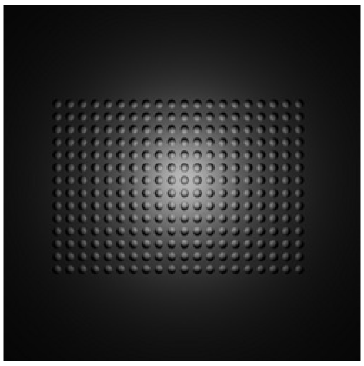
When the Quadratic slider is set to **0.0**, the formula for working out the attenuation at a particular range for full linear attenuation is:

```
I = E × (D / (D + L × r))
```

Where

- `I` is the calculated Intensity of light.
- `E` is the current Energy slider setting.
- `D` is the current setting of the Dist field.
- `L` is the current setting of the Linear slider.
- `r` is the distance from the lamp where the light intensity gets measured.

Quadratic


Lamp with Lin/Quad Weighted falloff default settings

This slider input field can have a value between **0.0** and **1.0**. A value of **1.0** in the Quadratic field and **0.0** in the Linear field means that the light from this source is completely quadratic.

Quadratic attenuation type lighting is considered a more accurate representation of how light attenuates (in the real world). In fact, fully quadratic attenuation is selected by default for Lin/Quad Weighted lamp fallout (see *Lamp with Lin/Quad Weighted falloff default settings*).

Here again, the light intensity is half when it reaches the Distance value from the lamp. Comparing the quadratic falloff to the linear falloff, the intensity decays much slower at distances lower than the set Distance, but it attenuates much quicker after Distance is reached.

When the Linear slider is set to **0.0**, the formula for working out the attenuation at a particular range for full quadratic attenuation is:

$I = E \times (D^2 / (D^2 + Q \times r^2))$

Where

- `I` is the calculated Intensity of light.
- `E` is the current Energy slider setting.
- `D` is the current setting of the Dist field.
- `Q` is the current setting of the Quad slider.
- `r` is the distance from the lamp where the light intensity gets measured.

Mixing “Linear” and “Quad”

If both the Linear and Quad slider fields have values greater than **0.0**, then the formula used to calculate the light attenuation profile changes to this:

$I = E \times (D / (D + L \times r)) \times (D^2 / (D^2 + Q \times r^2))$

Where

- `I` is the calculated Intensity of light.
- `E` is the current Energy slider setting.
- `D` is the current setting of the Dist field.
- `L` is the current setting of the Linear slider.
- `Q` is the current setting of the Quad slider.
- `r` is the distance from the lamp where the light intensity gets measured.

Zeroing both “Linear” and “Quad”

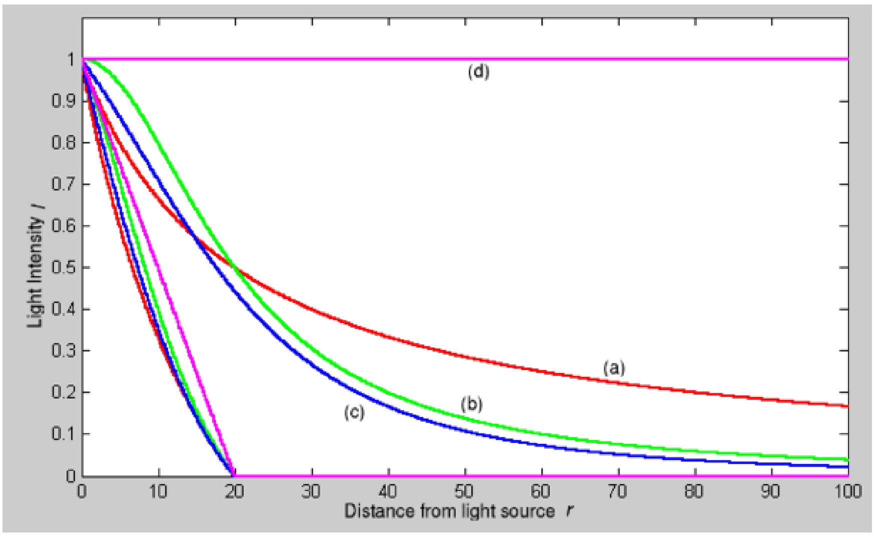
If both the Linear and Quadratic sliders have **0.0** as their values, the light intensity will not attenuate with distance. This does not mean that the light will not get darker—it will, but only because the energy the light has is spread out over a wider and wider distance. The total amount of energy in the spread-out light will remain the same, though. The light angle also affects the amount of light you see. It is in fact the behavior of light in the deep space vacuum.

If what you want is a light source that doesn’t attenuate and gives the same amount of light intensity to each area it hits, you need a light with properties like the Constant lamp Falloff type.

Also, when the Linear and Quad sliders are both **0.0** values the Distance field ceases to have any influence on the light attenuation, as shown by the equation above.

Graphical Summary

Below is a graph summarizing the lin/quad attenuation type, showing attenuation with or without the Sphere option (described later).



Light Attenuation:
a) Linear (Linear=**1.0**, Quad=**0.0**); b) Quadratic (Linear=**0.0**, Quad=**1.0**);
c) Linear and quadratic (Linear=Quad=**0.5**); d) Null (Linear=Quad=**0.0**).
Also shown in the graph the "same" curves, in the same colors, but with the Sphere button turned on.

Custom Curve

The Custom Curve lamp Falloff type is very flexible.

Most other lamp falloff types work by having their light intensity start at its maximum (when nearest to the light source) and then with some predetermined pattern decrease their light intensity when the distance from the light source increases.

When using the Custom Curve Lamp Falloff type, a new panel is created called Falloff Curve. This Falloff Curve profile graph allows the user to alter how intense light is at a particular point along a light's attenuation profile (i.e. at a specific distance from the light source).

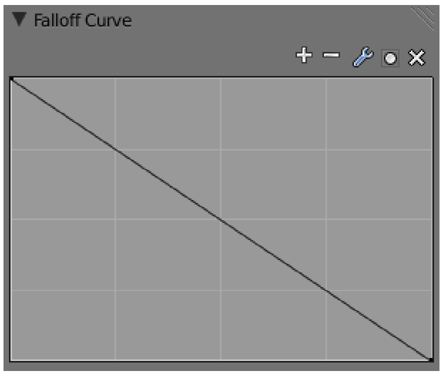
The Falloff Curve profile graph has two axes, the "`Distance`" axis and the "`Intensity`" axis.

Distance axis
: It represents the position at a particular point along a light source's attenuation path. The far left is at the position of the light source and the far right is the place where the light source's influence would normally be completely attenuated. I say "normally would" because the Falloff Curve can be altered to do the exact opposite if required.

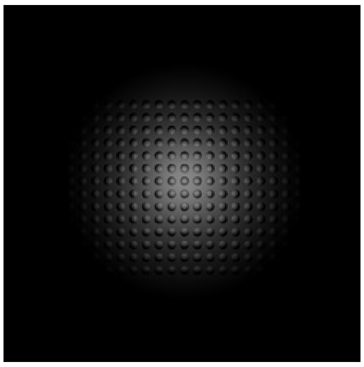
Intensity axis
: It represents the intensity at a particular point along a light source's attenuation path. Higher intensity is represented by being higher up the intensity axis, while lower intensity light is represented by being lower down on the intensity axis.

Altering the Falloff Curve profile graph is easy. Just LMB click on a part of the graph you want to alter and drag it where you want it to be. If when you click you are over or near one of the tiny black square handles, it will turn white, indicating that this handle is now selected, and you will be able to drag it to a new position. If when you click on the graph you are not near a handle, one will be created at the point that you clicked, which you can then drag where you wish. You can also create handles at specific parts of the graph, clicking with LMB while holding Ctrl key; it will create a new handle at the point you have clicked.

In the example below (the default for the Falloff Curve Profile Graph), the graph shows that the intensity of the light starts off at its maximum (when near the light), and linearly attenuates as it moves to the right (further away from the light source).

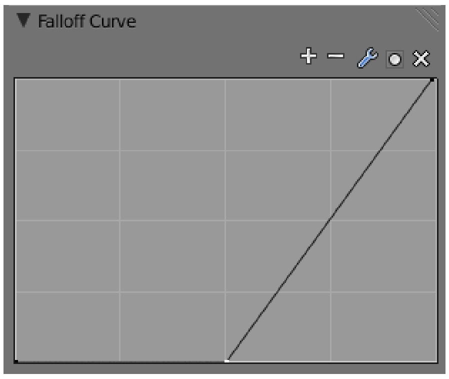


Default Falloff Curve panel graph.

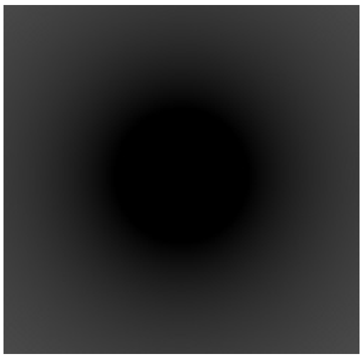


Render showing the Custom Curve lamp falloff type effect with default settings.

If you want to have a light attenuation profile that gets more intense as it moves away from the light source, you could alter the graph as below:

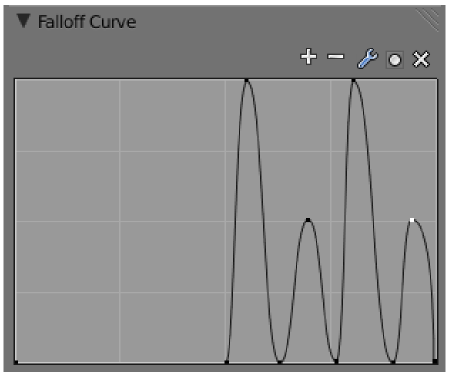


Falloff Curve for reversed attenuation.

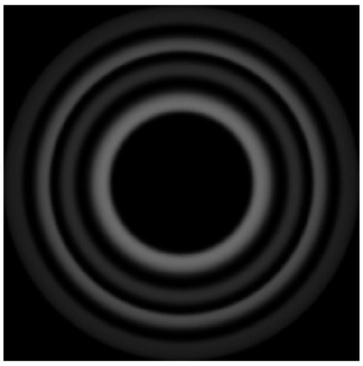

Falloff Curve for reversed attenuation rendered.

You are obviously not just limited to simple changes such as reversing the attenuation profile, you can have almost any profile you desire.

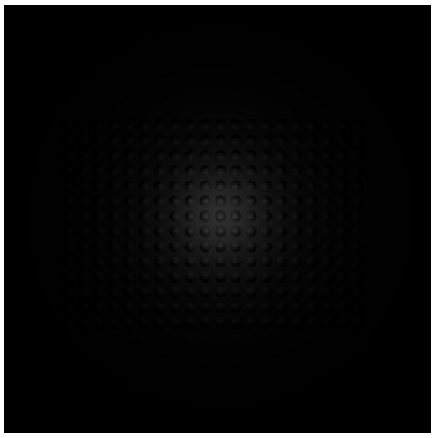
Here is another example of a different Falloff Curve profile graph, along with its resultant render output:



Oscillating attenuation profile.

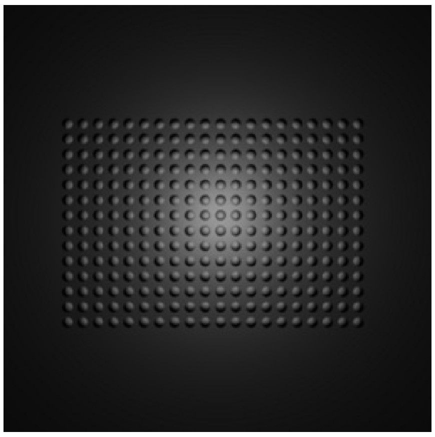

Render showing the effects of a “wavelet” profile graph on the light attenuation.

Inverse Square


Render showing the Inverse Square lamp falloff type effect with default settings.

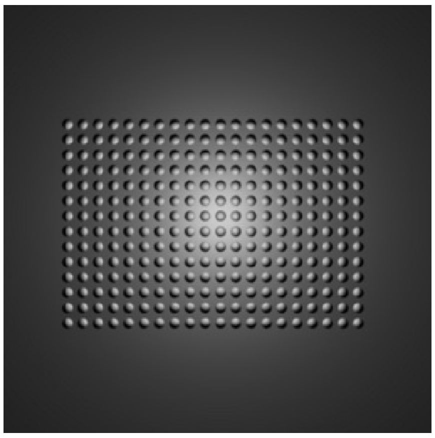
This lamp falloff type attenuates its intensity according to inverse square law, scaled by the Distance value. Inverse square is a sharper, realistic decay, useful for lighting such as desk lamps and street lights. This is similar to the old Quad option (and consequently, to the new Lin/Quad Weighted option with Linear to **0.0** and Quad to **1.0**), with slight changes.

Inverse Linear


Render showing the Inverse Linear lamp falloff type effect with default settings.

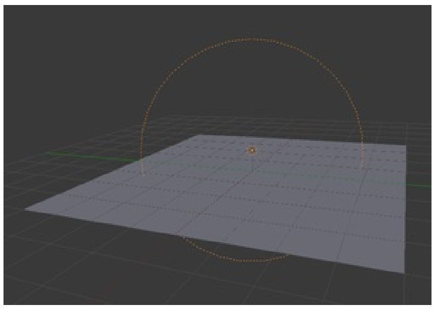
This lamp falloff type attenuates its intensity linearly, scaled by the Dist value. This is the default setting, behaving the same as the default in previous Blender versions without Quad switched on, and consequently, like the new Lin/Quad Weighted option with Linear to **1.0** and Quad to **0.0**. This isn't physically accurate, but can be easier to light with.

Constant


Render showing the Constant lamp falloff type effect with default settings.

This lamp falloff type does not attenuate its intensity with distance. This is useful for distant light sources like the sun or sky, which are so far away that their falloff isn't noticeable. Sun and Hemi lamps always have constant falloff.

Sphere


Screenshot of the 3D view window, showing the Sphere light clipping circle.

The Sphere option restricts the light illumination range of a Lamp or Spot lamp, so that it will completely stop illuminating an area once it reaches the number of Blender Units away from the Lamp, as specified in the Dist field.

When the Sphere option is active, a dotted sphere will appear around the light source, indicating the demarcation point at which this light intensity will be null.

The Sphere option adds a term to the chosen attenuation law, whatever it is:

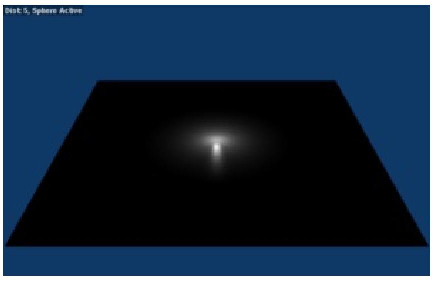
```
I' = I × (D - r) / D if r < D; 0 otherwise
```

Where:

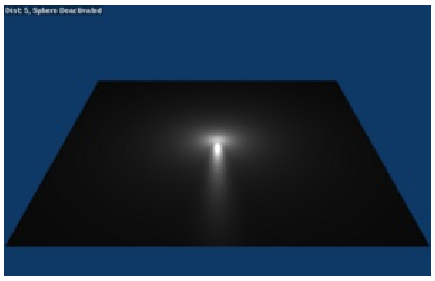
- `I'` is the required Intensity of light (with the Sphere option activated).
- `I` is the intensity of light calculated by the chosen attenuation law (without the Sphere option).
- `D` is the current setting of the Dist field.

- `r` is the distance from the lamp where the light intensity gets measured.

See the graphic at the end of the description of the Lin/Quad Weighted attenuation option.



Render showing the light attenuation of a Constant falloff light type with the Sphere option active.

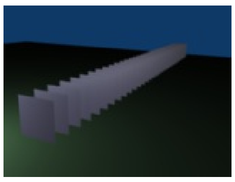


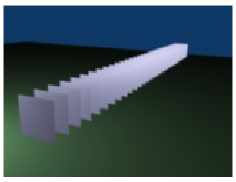
Render showing the light attenuation of a Constant falloff light type with the Sphere option deactivated.

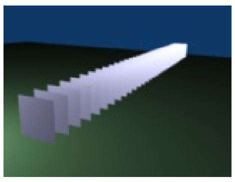
Examples

Distance

In this example, the Lamp has been set pretty close to the group of planes. This causes the light to affect the front, middle and rear planes more dramatically. Looking at (*Various Distance settings*), you can see that as the Dist is increased, more and more objects become progressively brighter.


Distance: **10**.


Distance: **100**.


Distance: **1000**.

Various Distance settings (shadows disabled).

The Distance parameter is controlling where the light is falling – at a linear rate by default – to half its original value from the light's origin. As you increase or decrease this value, you are changing where this half falloff occurs. You could think of Distance as the surface of a sphere and the surface is where the light's intensity has fallen to half its strength in all directions. Note that the light's intensity continues to fall even after Distance. Distance just specifies the distance where half of the light's energy has weakened.

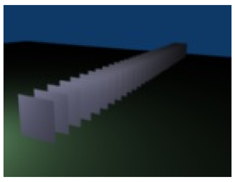
Notice in (*Distance: **1000***) that the farthest objects are very bright. This is because the falloff has been extended far into the distance, which means the light is very strong when it hits the last few objects. It is not until **1000** units that the light's intensity has fallen to half of its original intensity.

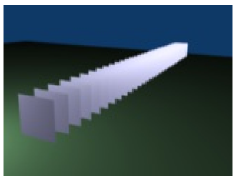
Contrast this with (*Distance: **10***), where the falloff occurs so soon that the farther objects are barely lit. The light's intensity has fallen by a half by time it even reaches the tenth object.

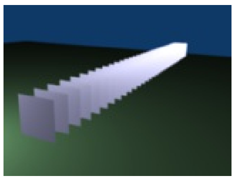
You may be wondering why the first few planes appear to be dimmer? This is because the surface angle between the light and the object's surface normal is getting close to oblique. That is the nature of a Lamp light object. By moving the light infinitely far away you would begin to approach the characteristics of the Sun lamp type.

Inverse Square

Inverse Square makes the light's intensity falloff with a non-linear rate, or specifically, a quadratic rate. The characteristic feature of using Inverse Square is that the light's intensity begins to fall off very slowly but then starts falling off very rapidly. We can see this in the (*Inverse Square selected*) images.


Inverse Square with **10**.


Inverse Square with **100**.


Inverse Square with **1000**.

Inverse Square selected (with the specified distances).

With Inverse Square selected, the Distance field specifies where the light begins to fall off faster, roughly speaking; see the light attenuation description for more info.

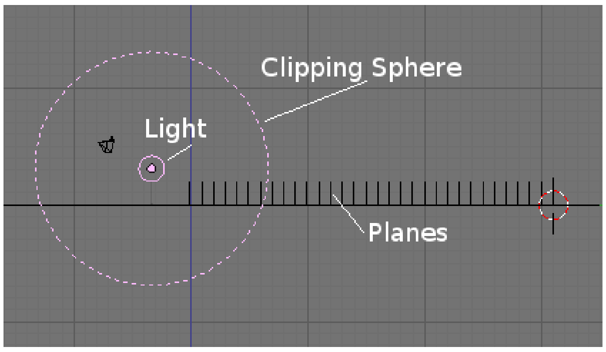
In (*Inverse Square with **10***), the light's intensity has fallen so quickly that the last few objects aren't even lit.

Both (*Inverse Square with **100***) and (*Inverse Square with **1000***) appear to be almost identical and that is because the Distance is set beyond the farthest object's distance which is at about **40 BU** out. Hence, all the objects get almost the full intensity of the light.

As above, the first few objects are dimmer than farther objects because they are very close to the light. Remember, the brightness of an object’s surface is also based on the angle between the surface normal of an object and the ray of light coming from the lamp.

This means there are at least two things that are controlling the surface’s brightness: intensity and the angle between the light source and the surface’s normal.

Sphere

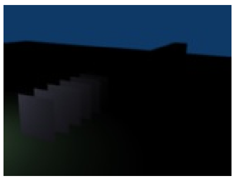
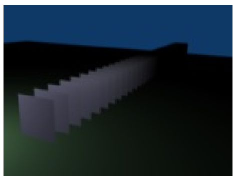
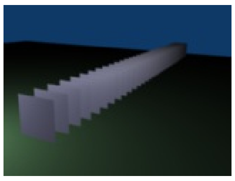


Clipping Sphere.

Sphere indicates that the light’s intensity is null at the Distance distance and beyond, regardless of the chosen light’s falloff. In (*Clipping Sphere*) you can see a side view example of the setup with Sphere enabled and a distance of **10**.

Any objects beyond the sphere receive no light from the lamp.

The Distance field is now specifying both where the light’s rays become null, and the intensity’s ratio falloff setting. Note that there is no abrupt transition at the sphere: the light attenuation is progressive (for more details, see the descriptions of the Sphere options and light attenuations above).


Sphere with **10**.

Sphere with **20**.

Sphere with **40**.

Sphere enabled with the specified distances, Inverse Linear light falloff.

In (*Sphere with **10***), the clipping sphere’s radius is **10** units, which means the light’s intensity is also being controlled by **10** units of distance. With a linear attenuation, the light’s intensity has fallen very low even before it gets to the first object.

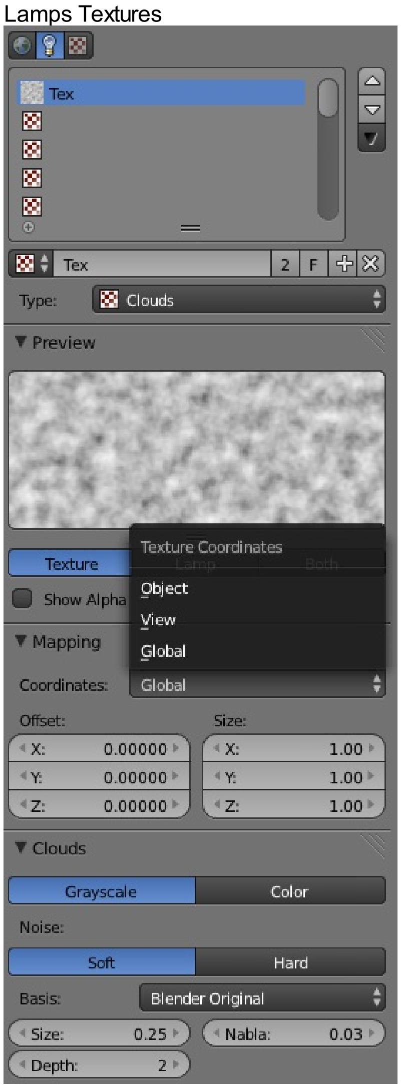
In (*Sphere with **20***), the clipping sphere’s radius is now **20 BU** and some light is reaching the middle objects.

In (*Sphere with **40***), the clipping sphere’s radius is now **40** units, which is beyond the last object. However, the light doesn’t make it to the last few objects because the intensity has fallen to nearly **0**.

Hints

If a Lamp light is set to not cast shadows, it illuminates through walls and the like. If you want to achieve some nice effects like a fire, or a candle-lit room interior seen from outside a window, the Sphere option is a must. By carefully working on the Distance value you can make your warm firelight shed only within the room, while illuminating outside with a cool moonlight, the latter achieved with a Sun or Hemi light or both.

Lamps Textures



Lamp Texture panels

When a new lamp is added, it produces light in a uniform, flat color. While this might be sufficient in simple renderings, more sophisticated effects can be accomplished through the use of textures. Subtle textures can add visual nuance to a lamp, while hard textures can be used to simulate more pronounced effects, such as a disco ball, dappled sunlight breaking through treetops, or even a projector. These textures are assigned to one of ten channels, and behave exactly like material textures, except that they affect a lamp's color and intensity, rather than a material's surface characteristics.

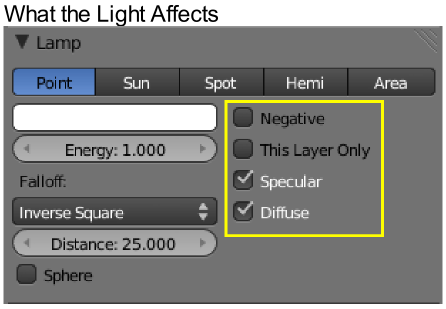
Options

The lamp textures settings are grouped into two panels. Here we will only talk about the few things that differ from object material textures; see the Materials and Textures chapters for details about the standard options.

The texture-specific and the Mapping panels remain the same. However, you'll note there are much fewer Mapping options – you can only choose between Global, View or another Object's texture coordinates (since a lamp has no texture coordinates by itself), and you can scale or offset the texture.

The Mapping panel is also a subset of its regular material's counterpart. You can only map a lamp texture to its regular, basic Color and/or to its Shadow color. As you can only affect colors, and a lamp has no texture coordinates on its own, the Diffuse, Specular, Shading, and Geometry options have disappeared.

What the Light Affects



Lamp panel with the light affecting options highlighted

Every lamp has a set of switches that control which objects receive its light, and how it interacts with materials.

Negative
: The light produced by the lamp is **subtracted** from the one “available” on the surfaces it hits, which darkens these surfaces instead of brightening them.

This Layer Only
: Causes the lamp to only light objects on the same layer.

Diffuse
: Prevents the lamp from producing diffuse light (it doesn’t really “light” things).

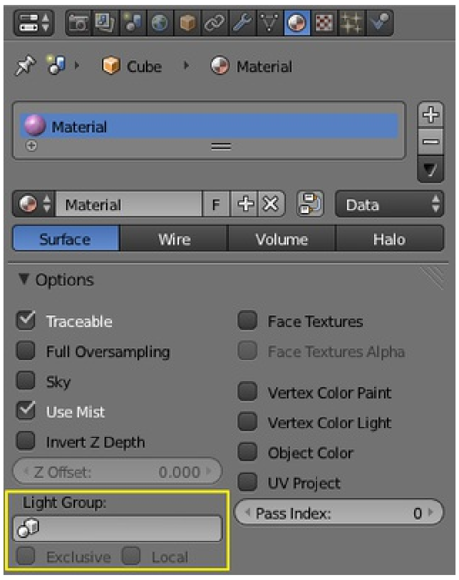
Specular
: Prevents the lamp from producing specular highlights.

Lamps Related Settings

Here are some options closely related to light sources, without being lamps settings.

Lighting Groups

Materials



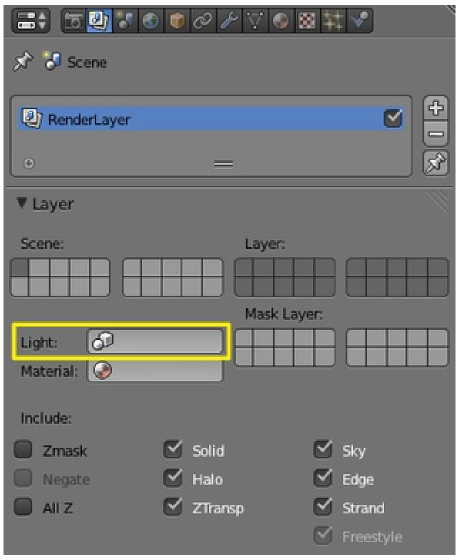
- [✓] Traceable
- [ ] Full Oversampling
- [ ] Sky
- [✓] Use Mist
- [ ] Invert Z Depth
- Z Offset: 0.000
- Light Group:
- [ ] Exclusive [ ] Local
- [ ] Face Textures
- [ ] Face Textures Alpha
- [ ] Vertex Color Paint
- [ ] Vertex Color Light
- [ ] Object Color
- [ ] UV Project
- Pass Index: 0

Light Group options for Materials

By default, materials are lit by all lamps in all visible layers, but a material (and thus all objects using that material) can be limited to a single group of lamps. This sort of control can be incredibly useful, especially in scenes with complex lighting setups. To enable this, navigate to the Material menu's Options panel and select a group of lamps in the Light Group field. Note that a light group must be created first.

If the Exclusive button is enabled, lights in the specified group will *only* affect objects with this material.

Render Layers



- Light:
- Material:
- Include:
- [ ] Zmask
- [ ] Negate
- [ ] All Z
- [✓] Solid
- [✓] Halo
- [✓] ZTransp
- [✓] Sky
- [✓] Edge
- [✓] Strand
- [✓] Freestyle

Light Group options for Render Layers

There's a similar control located in the Layer panel of the context Render Layers. If a light group name is selected in this Light field, the scene will be lit exclusively by lamps in the specified group.

See Also

- Lamps Introduction
- Shadows
- Materials Introduction

Shadows

Light wouldn’t even exist without its counterpart: shadows. Shadows are a darkening of a portion of an object because light is being partially or totally blocked from illuminating the object. They add contrast and volume to a scene; there is nearly no place in the real world without shadows, so to get realistic renders, you will need them. Blender supports the following kinds of shadows:

1. Ray-traced lamp shadows
2. Buffered lamp shadows
3. Ambient occlusion
4. Indirect lighting

Ambient occlusion really isn’t a shadow based on light *per se*, but based on geometry. However, it does mimic an effect where light is prevented from fully and uniformly illuminating an object, so it is mentioned here. Also, it is important to mention ambient lighting, since increasing Ambient decreases the effect of a shadow.

You can use a combination of ray-traced and buffer shadows to achieve different results. Even within ray-traced shadows, different lamps cast different patterns and intensities of shadow. Depending on how you arrange your lamps, one lamp may wipe out or override the shadow cast by another lamp.

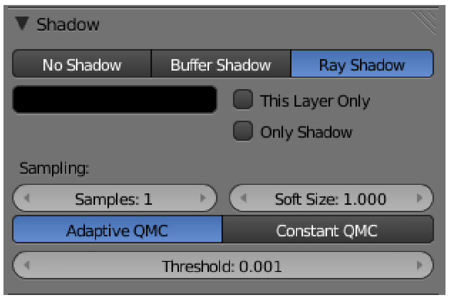
Shadows is one of those trifectas in Blender, where multiple things have to be set up in different areas to get results:

1. The lamp has to cast shadows (ability and direction).
2. An opaque object has to block light on its way (position and layer).
3. Another object’s material has to receive shadows (Shadow and Receive Transparent enabled).
4. The render engine has to calculate shadows (Shadow for buffered shadows, Shadow and Ray for ray-traced shadows).

For example, the simple Lamp, Area, and Sun light has the ability to cast ray shadows, but not buffer shadows. The Spot light can cast both, whereas the Hemi light does not cast any. If a Sun lamp is pointing sideways, it will not cast a shadow from a sphere above a plane onto the plane, since the light is not traveling that way. All lamps able to cast shadows share some common options, described here.

Just to give you more shadow options (and further confuse the issue), lamps and materials can be set to respectively **only** cast and receive shadows, and not light the diffuse/specular aspects of the object. Also, render layers can turn on/off the shadow pass, and their output may or may not contain shadow information…

Lamps: Ray-traced Shadows



Ray Shadow enabled for a lamp

Ray-traced shadows produce very precise shadows with very low memory use, but at the cost of processing time. This type of shadowing is available to all lamp types except Hemi.

As opposed to buffered shadows, ray-traced shadows are obtained by casting rays from a regular light source, uniformly and in all directions. The ray-tracer then records which pixel of the final image is hit by a ray light, and which is not. Those that are not are obviously obscured by a shadow.

Each light casts rays in a different way. For example, a Spot light casts rays uniformly in all directions within a cone. The Sun light casts rays from a infinitely distant point, with all rays parallel to the direction of the Sun light.

For each additional light added to the scene, with ray-tracing enabled, the rendering time increases. Ray-traced shadows require more computation than buffered shadows but produce sharp shadow borders with very little memory resource usage.

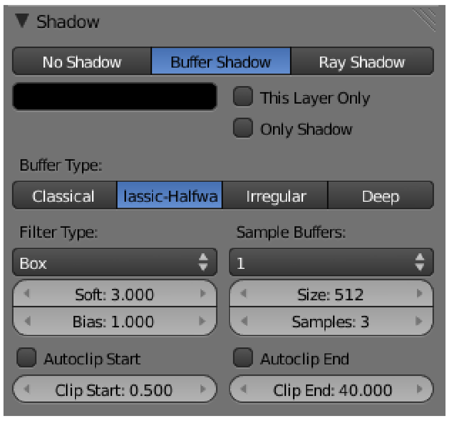
To enable ray-traced shadows, three actions are required:

- Enable Shadows globally in the Render menu's Shading panel.
- Enable Ray tracing globally from the same panel.
- Enable ray-traced shadows for the light using the Ray Shadow button in the Light menu's Shadow panel. This panel varies depending on the type of light.
    - All lamps able to cast ray-traced shadows share some common options, described in Ray-traced Properties.

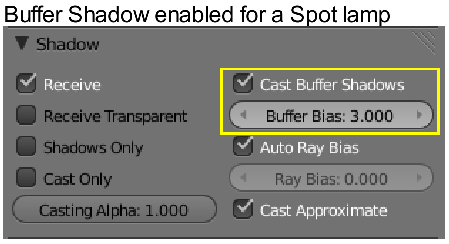
Ray-traced shadows can be cast by the following types of lamp:

- Point lamp
- Spot lamp
- Area lamp
- Sun lamp

Lamps: Buffered Shadows



Buffer Shadow enabled for a Spot lamp



Cast Buffer Shadows enabled for a material

Buffered shadows provide fast-rendered shadows at the expense of precision and/or quality. Buffered shadows also require more memory resources as compared to ray tracing. Using buffered shadows depends on your requirements. If you are rendering animations or can't wait hours to render a complex scene with soft shadows, buffer shadows are a good choice.

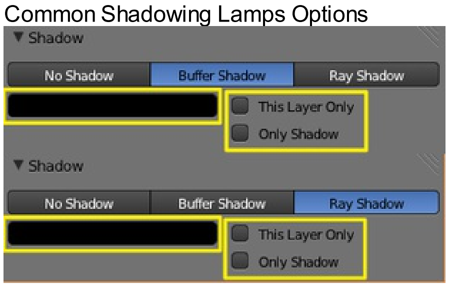
For a scanline renderer – and Blender's built-in engine *is*, among other things, a scanline renderer – shadows can be computed using a *shadowbuffer*. This implies that an "image", as seen from the spot lamp's point of view, is "rendered" and that the distance – in the image – for each point from the spot light is saved. Any point in the "rendered" image that is farther away than any of those points in the spot light's image is then considered to be in shadow. The shadow buffer stores this image data.

To enable buffered shadows these actions are required:

- Enable shadows globally from the Scene menu's Gather panel by selecting Approximate.
- Enable shadows for the light using the Buffer Shadow button in the Lamp menu's Shadow panel.
- Make sure the Cast Buffer Shadows options is enabled in each Material's Shadow panel.

- The Spot lamp is the only lamp able to cast buffered shadows.

Common Shadowing Lamps Options



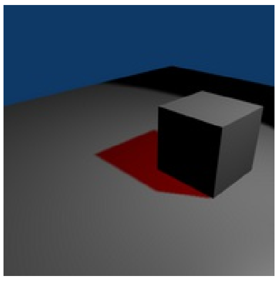
Common shadowing options for lamps

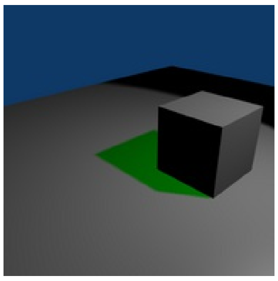
All lamps able to cast shadows (Lamp, Spot, Area and Sun) share some options, described below:

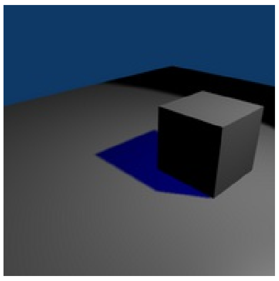
This Layer Only
: When this option is enabled, only the objects on the same layer as the light source will cast shadows.

Only Shadow
: The light source will not illuminate an object but will generate the shadows that would normally appear.
This feature is often used to control how and where shadows fall by having a light which illuminates but has no shadow, combined with a second light which doesn’t illuminate but has Only Shadow enabled, allowing the user to control shadow placement by moving the “Shadow Only” light around.

Shadow color
: This color picker control allows you to choose the color of your cast shadows (black by default).
The images below were all rendered with a white light and the shadow color was selected independently.


Red colored shadow example


Green colored shadow example

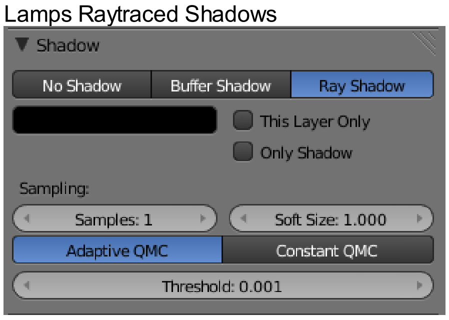

Blue colored shadow example

Although you can select a pure white color for a shadow color, it appears to make a shadow disappear.

See Also

- Shadows
- Common Raytraced Options
- Lamp Light Raytraced Shadows
- Spot Light Raytraced Shadows
- Area Light Raytraced Shadows
- Sun Light Raytraced Shadows
- Spot Light Buffered Shadows

Lamps Raytraced Shadows


Shadow

No Shadow | Buffer Shadow | Ray Shadow

This Layer Only

Only Shadow

Sampling:

Samples: 1

Soft Size: 1.000

Adaptive QMC | Constant QMC

Threshold: 0.001

Ray shadowing options for lamps

Most lamp types (Lamp, Spot and Sun) share the same options for the ray-traced shadows generation, which are described below. Note that the Area lamp, even though using most of these options, have some specifics described in its own ray-traced shadows page.

Ray Shadow
: The Ray Shadow button enables the light source to generate ray-traced shadows.
: When the Ray Shadow button is selected, another set of options is made available, those options being:

Shadow sample generator type
: Method for generating shadow samples: Adaptive QMC is fastest, Constant QMC is less noisy but slower.
: This allows you to choose which algorithm is to be used to generate the samples that will serve to compute the ray-traced shadows (for now, mainly two variants of Quasi-Monte Carlo, see below):

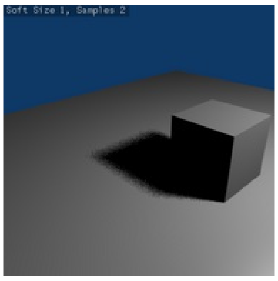
  Constant QMC
  : The Constant QMC method is used to calculate shadow values in a very uniform, evenly distributed way. This method results in very good calculation of shadow value but it is not as fast as using the Adaptive QMC method; however, Constant QMC is more accurate.

  Adaptive QMC
  : The Adaptive QMC method is used to calculate shadow values in a slightly less uniform and distributed way. This method results in good calculation of shadow value but not as good as Constant QMC. The advantage of using Adaptive QMC is that it is in general much quicker while being not much worse than Constant QMC in terms of overall results.

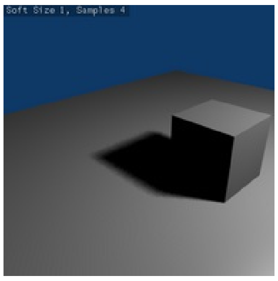
Samples
: Number of extra samples taken (samples x samples).
: This slider sets the maximum number of samples that both Constant QMC and Adaptive QMC will use to do their shadow calculations. The maximum value is **16** – the real number of samples is actually the square of it, so setting a sample value of **3** really means $3^2 = 9$ samples will be taken.

Soft Size
: Light size for ray shadow sampling. This slider determines the size of the fuzzy/diffuse/penumbra area around the edge of a shadow. Soft Size only determines the width of the soft shadow size, not how graded and smooth the shadow is. If you want a wide shadow which is also soft and finely graded, you must also set the number of samples in the Samples field higher than **1**; otherwise this field has no visible effect and the shadows generated will not have a soft edge. The maximum value for Soft Size is **100.0**.

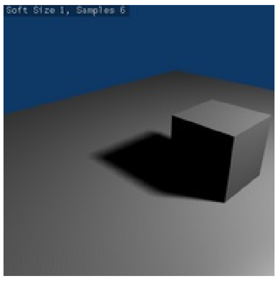
  Below is a table of renders with different Soft Size and Samples settings showing the effect of various values on the softness of shadow edges:



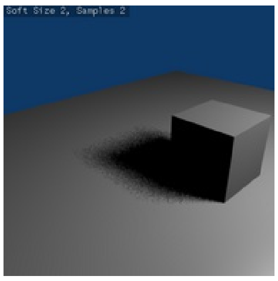
Soft Size: **1.0**, Samples: **2**.



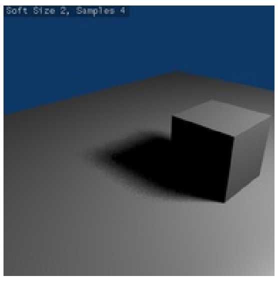
Soft Size: **1.0**, Samples: **4**.



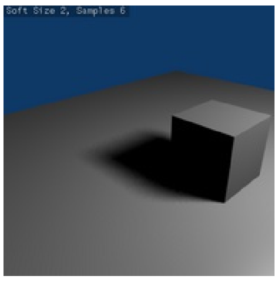
Soft Size: **1.0**, Samples: **6**.



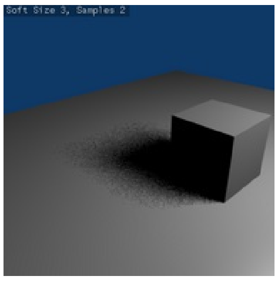
Soft Size: **2.0**, Samples: **2**.



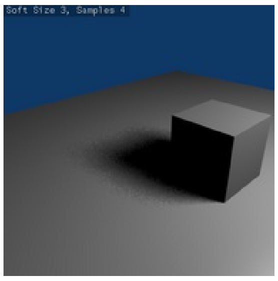
Soft Size: **2.0**, Samples: **4**.



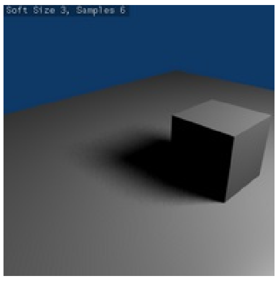
Soft Size: **2.0**, Samples: **6**.



Soft Size: **3.0**, Samples: **2**.



Soft Size: **3.0**, Samples: **4**.



Soft Size: **3.0**, Samples: **6**.

Below is an animated version of the above table of images showing the effects:

Animated version renders with different Soft Size and Samples settings showing the effect of various values on the softness of shadow edges.

Threshold
: Threshold for Adaptive Sampling. This field is used with the Adaptive QMC shadow calculation method. The value is used to determine if the Adaptive QMC shadow sample calculation can be skipped based on a threshold of how shadowed an area is already. The maximum Threshold value is **1.0**.

What is Quasi-Monte Carlo?

The Monte Carlo method is a method of taking a series of samples/readings of values (any kind of values, such as light values, color values, reflective states) in or around an area at random, so as to determine the correct actions to take in certain calculations which usually require multiple sample values to determine overall accuracy of those calculations. The Monte Carlo method tries to be as random as possible; this can often cause areas that are being sampled to have large irregular gaps in them (places that are not sampled/read). This in turn can cause problems for certain calculations (such as shadow calculation).

The solution to this was the Quasi-Monte Carlo method.

The Quasi-Monte Carlo method is also random, but tries to make sure that the samples/readings it takes are also better distributed (leaving less irregular gaps in its sample areas) and more evenly spread across an area. This has the advantage of sometimes leading to more accurate calculations based on samples/reading.

Volumetric Lighting

According to Wikipedia, volumetric lighting

« *is a technique used in 3D computer graphics to add lighting effects to a rendered scene. It allows the viewer to see beams of light shining through the environment; seeing sunbeams streaming through an open window is an example of volumetric lighting, also known as God rays. The term seems to have been introduced from cinematography and is now widely applied to 3D modeling and rendering especially in the field of 3D gaming.*
In volumetric lighting, the light cone emitted by a light source is modeled as a transparent object and considered as a container of a "volume": as a result, light has the capability to give the effect of passing through an actual three dimensional medium (such as fog, dust, smoke, or steam) that is inside its volume, just like in the real world. »

A classic example is the search light with a visible halo/shaft of light being emitted from it as the search light sweeps around.

By default Blender does not model this aspect of light. For example when Blender lights something with a Spot light, you see the objects and area on the floor lit but not the shaft/halo of light coming from the spotlight as it progresses to its target and would get scattered on the way.

The halo/shaft of light is caused in the real world by light being scattered by particles in the air, some of which get diverted into your eye and that you perceive as a halo/shaft of light. The scattering of light from a source can be simulated in Blender using various options, but by default is not activated.

The only lamp able to create volumetric effects is the Spot lamp (even thought you might consider some of the "Sky & Atmosphere" effects of the Sun lamp as volumetric as well).

Example

Blend file of spotlight animation.
[video link]

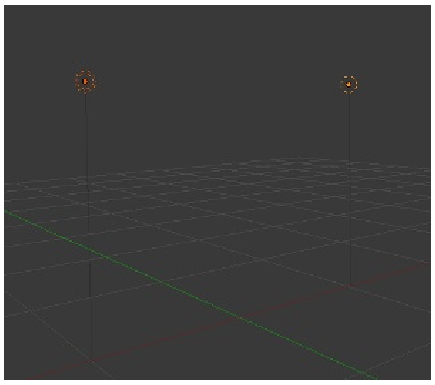
See also

- Mist
- Smoke
- Volumetric Materials

Lamps

Blender comes equipped with five different lamp types, each with its own unique strengths and limitations. Here are the available lamps:

- Point is an omni-directional point light source, similar to a light bulb.
- Spot is a directional point light source, similar to … a spot.
- Area is a source simulating an area which is producing light, as windows, neons, TV screens.
- Hemi simulates a very wide and far away light source, like the sky.
- Sun simulates a very far away and punctual light source, like the sun.


Visual height and shadow markers of two points lamps. Ray Shadow is enabled on the left lamp.

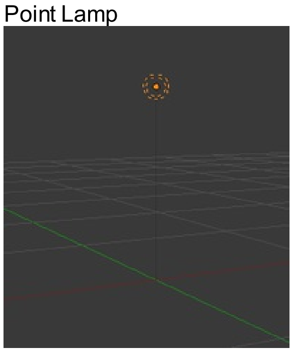
You can add new lamps to a scene using the Add menu in the top header, or with (⇧ ShiftA » Add » Lamp).

Once added, a lamp's position is indicated in the 3D View by a solid dot in a circle, but most types also feature dashed wire-frames that help describe their orientation and properties. While each type is represented differently, there are some visual indicators common to all of them:

Shadows
: If shadows are enabled, an additional dashed circle is drawn around the solid circle. This makes it easier to quickly determine if a lamp has shadows enabled.

Vertical Height Marker
: This is a dim gray line, which helps locate the lamp's position relative to the global X-Y plane.

Point Lamp


Point lamp

The Point lamp is an omni-directional point of light, that is, a point radiating the same amount of light in all directions. It's visualized by a plain, circled dot. Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the lamp and the point on the surface of the object itself.

Light intensity/energy decays based on (among other variables) distance from the Point lamp to the object. In other words, surfaces that are further away are rendered darker.

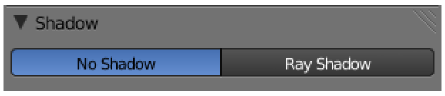
Lamp Options

Distance, Energy and Color
: These settings are common to most types of lamps, and are described in Light Properties.

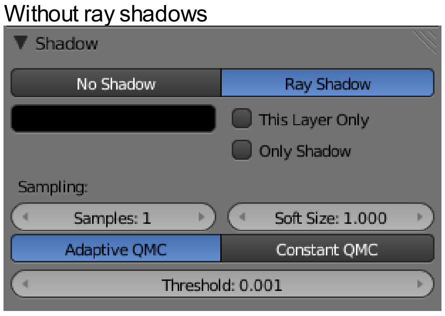
Negative, This Layer Only, Specular, and Diffuse
: These settings control what the lamp affects, as described in What Light Affects.

Falloff and Sphere
: These settings control how the light of the Lamp decays with distance. See Light Attenuation for details.

Shadows



Without ray shadows

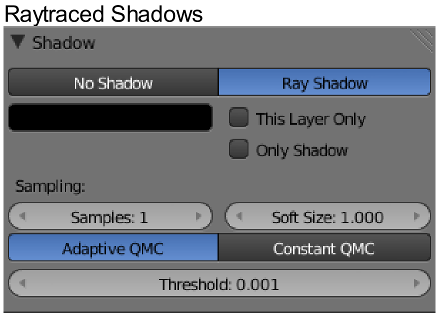


Point lamp with ray shadows and Adaptive QMC sample generator enabled

The Point light source can only cast ray-traced shadows. It shares with other lamp types the common shadow options described in Shadow Properties.

The ray-traced shadows settings of this lamp are shared with other lamps, and are described Raytraced Properties.

Raytraced Shadows



Shadow panel

The Point light source can only cast raytraced shadows. It shares with other lamp types the same common shadowing options, described in Shadows Properties.

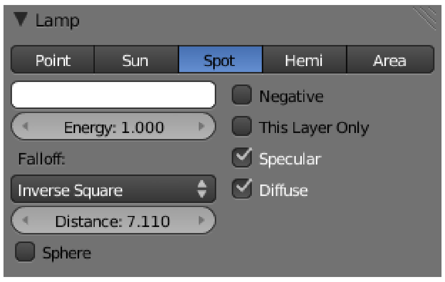
The raytraced shadows settings of this lamp are shared with other ones, and are described in Raytraced Properties.

Spot Lamp

A Spot lamp emits a cone-shaped beam of light from the tip of the cone, in a given direction.

The Spot light is the most complex of the light objects and indeed, for a long time, among the most used thanks to the fact that it was the only one able to cast shadows. Nowadays, with a ray tracer integrated into Blender's internal render engine, all lamps can cast shadows (except Hemi). Even so, Spot lamps' shadow buffers are much faster to render than ray-traced shadows, especially when blurred/softened, and spot lamps also provide other functionality such as "volumetric" halos.

Lamp options


Lamp

Point | Sun | Spot | Hemi | Area (Spot selected)

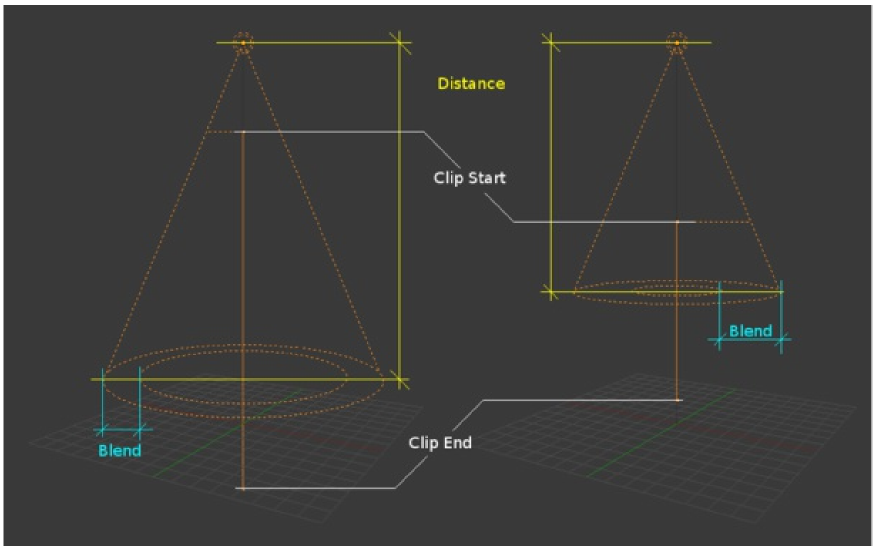
Negative: off
This Layer Only: off
Energy: 1.000
Falloff: Inverse Square
Specular: on
Diffuse: on
Distance: 7.110
Sphere: off

Common Lamp options of a Spot

Distance, Energy and Color
: These settings are common to most types of lamps, and are described in Light Properties.

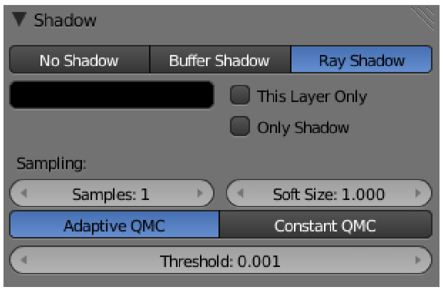
This Layer Only, Negative, Diffuse and Specular
: These settings control what the lamp affects, as described in What Light Affects.

Light Falloff and Sphere
: These settings control how the light of the Spot decays with distance. See Light Attenuation for details.



Changing the Spot options also changes the appearance of the spotlight as displayed in the 3D View

Shadows


Shadow

No Shadow | Buffer Shadow | Ray Shadow (Ray Shadow selected)

This Layer Only: off
Only Shadow: off
Sampling:
Samples: 1
Soft Size: 1.000
Adaptive QMC (selected) | Constant QMC
Threshold: 0.001

Shadow panel set to Ray Shadow

Spotlights can use either ray-traced shadows or buffered shadows. Either of the two can provide various extra options. Ray-traced shadows are generally more accurate, with extra capabilities such as transparent shadows, although they are quite slower to render.

No Shadow
: Choose this to turn shadows off for this spot lamp. This can be useful to add some discreet directed light to a scene.

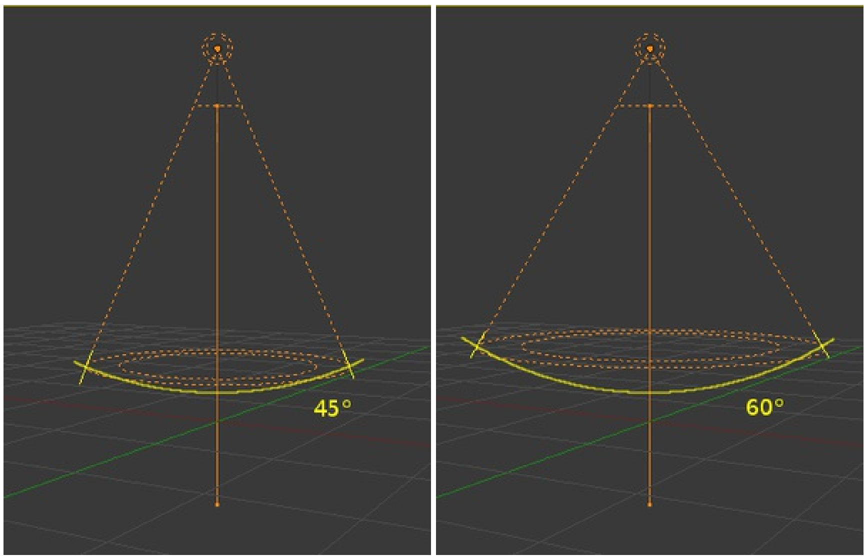
Buffer Shadow
: Buffered Shadows are also known as depth map shadows. Shadows are created by calculating differences in the distance from

the light to scene objects. See Buffered Shadows for full details on using this feature.
Buffered shadows are more complex to set up and involve more faking, but the speed of rendering is a definite advantage. Nevertheless, it shares with other lamp types common shadow options described in Shadows Properties.

Ray Shadow
: The ray-traced shadows settings of this lamp are shared with other lamps, and are described in Raytraced Properties.

Spot Shape

Size
: The size of the outer cone of a Spot, which largely controls the circular area a Spot light covers. This slider in fact controls the angle at the top of the lighting cone, and can be between **1.0°** and **180.0°**.



Changing the spot Size option

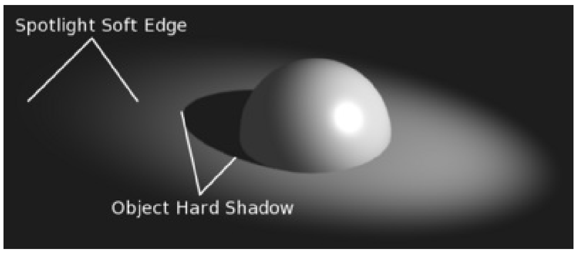
Blend
: The Blend slider controls the inner cone of the Spot. The Blend value can be between **0.0** and **1.0**. The value is proportional and represents that amount of space that the inner cone should occupy inside the outer cone (Size).

The inner cone boundary line indicates the point at which light from the Spot will start to blur/soften; before this point its light will mostly be full strength. The larger the value of Blend the more blurred/soft the edges of the spotlight will be, and the smaller the inner cone's circular area will be (as it starts to blur/soften earlier).

To make the Spot have a sharper falloff rate and therefore less blurred/soft edges, decrease the value of Blend. Setting Blend to **0.0** results in very sharp spotlight edges, without any transition between light and shadow.

The falloff rate of the Spot lamp light is a ratio between the Blend and Size values; the larger the circular gap between the two, the more gradual the light fades between Blend and Size.

Blend and Size only control the Spot light cone’s aperture and softness (“radial” falloff); they do not control the shadow’s softness as shown below.



Render showing the soft edge spotlighted area and the sharp/hard object shadow

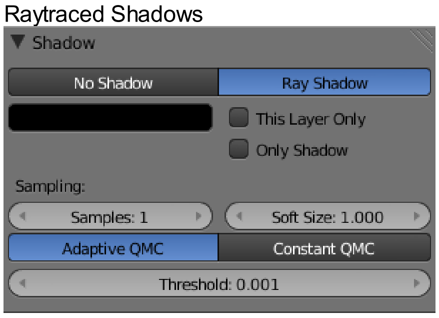
Notice in the picture above that the object’s shadow is sharp as a result of the ray tracing, whereas the spotlight edges are soft. If you want other items to cast soft shadows within the Spot area, you will need to alter other shadow settings.

Square
: The Square button makes a Spot light cast a square light area, rather than the default circular one.

Show Cone
: Draw a transparent cone in 3D view to visualize which objects are contained in it.

Halo
: Adds a volumetric effects to the spot lamp. See Spot Halos.

Raytraced Shadows

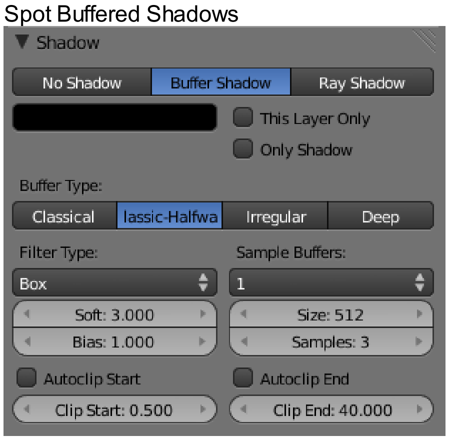


Shadow panel

The Spot light source can only cast raytraced shadows. It shares with other lamp types the same common shadowing options, described in Shadows Properties.

The raytraced shadows settings of this lamp are shared with other ones, and are described in Raytraced Properties.

Spot Buffered Shadows



Box
1
Soft: 3.000
Bias: 1.000
Size: 512
Samples: 3
Clip Start: 0.500
Clip End: 40.000

Buffer Shadow enabled for a Spot lamp

Spotlights can use either Raytraced Shadows or buffered shadows. Either of the two can provide various extra options.

Raytraced shadows are generally more accurate, with extra capabilities such as transparent shadows, although they are quite slower to render.

Buffered shadows are more complex to set up and involve more faking, but the speed of rendering is a definite advantage. Nevertheless, it shares with other lamp types common shadows options described in Shadows Properties.

Shadow Buffer Types

When the Buffer Shadow button is activated, the currently selected Spot light generates shadows, using a “shadow buffer” rather than using raytracing, and various extra options and buttons appear in the Shadow panel.

Buffer Type
: There more than one way to generate buffered shadows. The shadow buffer generation type controls which generator to use.

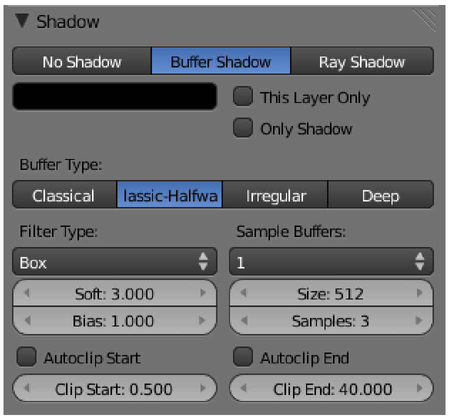
There are four shadow generation types, those being:

- Classical
- Classic-Halfway
- Irregular
- Deep

For more information on the different shadow generation methods see these links:

- Development Release Logs 2.43: Irregular Shadow Buffer
- Blender Nation: Blender Gets Irregular Shadow Buffers
- Development Release Logs 2.43: Shadow Buffer Halfway Average

“Classical” and “Classic-Halfway”



Box
1
Soft: 3.000
Bias: 1.000
Size: 512
Samples: 3
Clip Start: 0.500
Clip End: 40.000

Buffer Shadowset to Classic-Halfway

Classical
: A shadow generation which used to be the Blender default and unique method for generation of buffered shadows. It used an older way of generating buffered shadows, but it could have some problems with accuracy of the generated shadows and can be very sensitive to the resolution of the shadow buffer (Shadow Buffer→Size), different Bias values, and all the self-shadowing issues that brings up.

  The Classical method of generating shadows is obsolete and is really only still present to allow for backward compatibility with older versions of Blender. In most other cases you will want to use Classic-Halfway instead.

Classic-Halfway

This shadow buffer type is an improved shadow buffering method and is the default option selected in Blender. It works by taking an averaged reading of the first and second nearest Z depth values allowing the Bias value to be lowered and yet not suffer as much from self-shadowing issues.

Not having to increase Bias values helps with shadow accuracy, because large Bias values can mean small faces can lose their shadows, as well as preventing shadows being overly offset from the larger Bias value.

Classic-Halfway doesn't work very well when faces overlap, and biasing problems can happen.

Here are now the options specific to these generation methods:

Size
The Size numeric field can have a value from **512** to **10240**. Size represents the resolution used to create a shadow map. This shadow map is then used to determine where shadows lay within a scene.

As an example, if you have a Size with a value of **1024**, you are indicating that the shadow data will be written to a buffer which will have a square resolution of **1024×1024** pixels/samples from the selected spotlight.

The higher the value of Size, the higher resolution and accuracy of the resultant shadows, assuming all other properties of the light and scene are the same, although more memory and processing time would be used. The reverse is also true – if the Size value is lowered, the resultant shadows can be of lower quality, but would use less memory and take less processing time to calculate.

As well as the Size value affecting the quality of generated shadows, another property of Spot lamps that affects the quality of their buffered shadows is the angle of the spotlights lighted area (given in the Spot Shape panel's Size field).

As the spot shape Size value is increased, the quality of the cast shadows degrades. This happens because when the Spot lighted area is made larger (by increasing spot shape Size), the shadow buffer area has to be stretched and scaled to fit the size of the new lighted area.

The Size resolution is not altered to compensate for the change in size of the spotlight, so the quality of the shadows degrades. If you want to keep the generated shadows the same quality, as you increase the spot shape Size value, you also need to increase the buffer Size value.

The above basically boils down to
If you have a spotlight that is large you will need to have a larger buffer Size to keep the shadows good quality. The reverse is true also – the quality of the generated shadows will usually improve (up to a point) as the Spot lamp covers a smaller area.

Filter Type
The Box, Tent, and Gauss filter types control what filtering algorithm to use to anti-alias the buffered shadows.

They are closely related to the Samples numeric field, as when this setting is set to **1**, shadow filtering is disabled, so none of these buttons will have any effect what soever.

Box
The buffered shadows will be anti-aliased using the “box” filtering method.
This is the original filter used in Blender. It is relatively low quality and is used for low resolution renders, as it produces very sharp anti-aliasing. When this filter is used, it only takes into account oversampling data which falls within a single pixel, and doesn't take into account surrounding pixel samples. It is often useful for images which have sharply angled elements and horizontal/vertical lines.

Tent
The buffered shadows will be anti-aliased using the “tent” filtering method.
It is a simple filter that gives sharp results, an excellent general purpose filtering method. This filter also takes into account the sample values of neighboring pixels when calculating its final filtering value.

Gauss
The buffered shadows will be anti-aliased using the “Gaussian” filtering method.
It produces a very soft/blurry anti-aliasing. As result, this filter is excellent with high resolution renders.

The Anti-Aliasing page in the Render chapter will give more information on the various filtering/distribution methods and their uses.

Samples
The Samples numeric field can have a value between **1** and **16**. It controls the number of samples taken per pixel when calculating shadow maps.

The higher this value, the more filtered, smoothed and anti-aliased the shadows cast by the current lamp will be, but the longer they will take to calculate and the more memory they will use. The anti-aliasing method used is determined by having one of the Box, Tent or Gauss buttons activated (see above).

Having a Samples value of **1** is similar to turning off anti-aliasing for buffered shadows.

Soft
The Soft numeric field can have a value between **1.0** and **100.0**. It indicates how wide an area is sampled when doing anti-aliasing on buffered shadows. The larger the Soft value, the more graduated/soft the area that is anti-aliased/softened on the edge of generated shadows.

Sample Buffers
The Sample Buffers setting can be set to values **1**, **4** or **9**, and represents the number of shadow buffers that will be used when

doing anti-aliasing on buffered shadows.

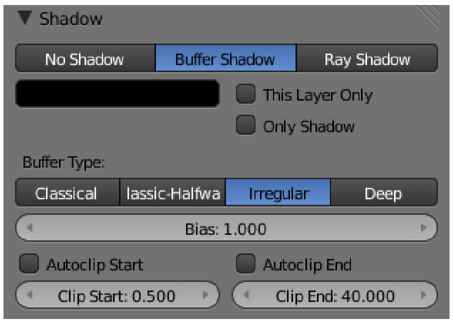
This option is used in special cases, like very small objects which move and need to generate really small shadows (such as strands). It appears that normally, pixel width shadows don't anti-alias properly, and that increasing Buffer Size doesn't help much.

So this option allows you to have a sort of extra sample pass, done above the regular one (the one controlled by the Box/Tent/Gauss, Samples and Soft settings).

The default **1** value will disable this option.

Higher values will produce a smoother anti-aliasing – but be careful: using a Sample Buffers of **4** will require four times as much memory and process time, and so on, as Blender will have to compute that number of sample buffers.

"Irregular"



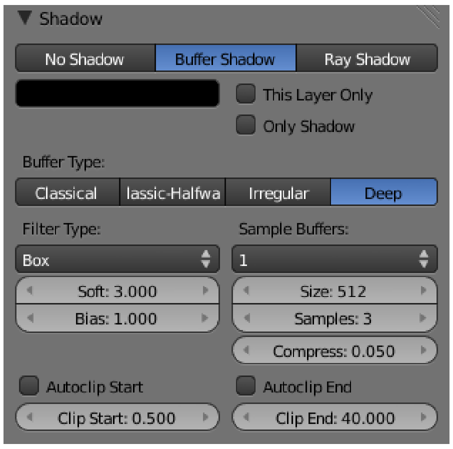
Bias: 1.000
Clip Start: 0.500
Clip End: 40.000

Buffer Shadow set to Irregular

Irregular shadow method is used to generate sharp/hard shadows that are placed as accurately as raytraced shadows. This method offers very good performance because it can be done as a multi-threaded process.

This method supports transparent shadows. To do so, you will first need to setup the shadow setting for the object which will receive the transparent shadow. (Material → Shadow → Cat Buffer Shadows and Buffer Bias)

Deep generation method



Filter Type: Box
Sample Buffers: 1
Soft: 3.000
Bias: 1.000
Size: 512
Samples: 3
Compress: 0.050
Clip Start: 0.500
Clip End: 40.000

Buffer Shadow set to Deep

Deep Shadow buffer supports transparency and better filtering , at the cost of more memory usage and processing time

Compress: Deep shadow map compression treshold

Common options

The following settings are common to all buffered shadow generation method.

Bias
: The Bias numeric field can have a value between **0.001** and **5.0**. Bias is used to add a slight offset distance between an object and the shadows cast by it. This is sometimes required because of inaccuracies in the calculation which determines weather an area of an object is in shadow or not.

Making the Bias value smaller results in the distance between the object and its shadow being smaller. If the Bias value is too small, an object can get artifacts, which can appear as lines and interference patterns on objects. This problem is usually called "self shadowing", and can usually be fixed by increasing the Bias value, which exists for that purpose!

Other methods for correcting self shadowing include increasing the size of the Shadow Buffer Size or using a different buffer shadow calculation method such as Classic-Halfway or Irregular.

Self shadowing interference tends to affect curved surfaces more than flat ones, meaning that if your scene has a lot of curved surfaces it may be necessary to increase the Bias value or Shadow Buffer Size value.

Having overly large Bias values not only places shadows further away from their casting objects, but can also cause objects that are very small to not cast any shadow at all. At that point altering Bias, Shadow Buffer Size or Spot Size values, among other things, may be required to fix the problem.

Finer Bias tuning
You can now refine the Bias value independently for each Material, using the Bias slider (Material menu, Shadow panel). This value is a factor by which the Bias value of each Spot buffered shadows lamp is multiplied, each time its light hits an object using this material. The **0.0** and **1.0** values are equivalent – they do not alter the lamp’s Bias original value.

Clip Start & Clip End
When a Spot light with buffered shadows is added to a scene, an extra line appears on the Spot 3D view representation.

The start point of the line represents Clip Start’s value and the end of the line represents Clip End’s value. Clip Start can have a value between **0.1** and **1000.0**, and Clip End, between **1.0** and **5000.0**. Both values are represented in Blender Units.

Clip Start indicates the point after which buffered shadows can be present within the Spot light area. Any shadow which could be present before this point is ignored and no shadow will be generated.

Clip End indicates the point after which buffered shadows will not be generated within the Spot light area. Any shadow which could be present after this point is ignored and no shadow will be generated.

The area between Clip Start and Clip End will be capable of having buffered shadows generated.

Altering the Clip Start and Clip End values helps in controlling where shadows can be generated. Altering the range between Clip Start and Clip End can help speed up rendering, save memory and make the resultant shadows more accurate.

When using a Spot lamp with buffered shadows, to maintain or increase quality of generated shadows, it is helpful to adjust the Clip Start and Clip End such that their values closely bound around the areas which they want to have shadows generated at. Minimizing the range between Clip Start and Clip End, minimizes the area shadows are computed in and therefore helps increase shadow quality in the more restricted area.

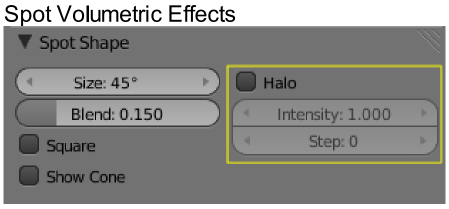
Autoclip Start & Autoclip End
As well as manually setting Clip Start and Clip End fields to control when buffered shadows start and end, it is also possible to have Blender pick the best value independently for each Clip Start and Clip End field.

Blender does this by looking at where the visible vertices are when viewed from the Spot lamp position.

Hints

Any object in Blender can act as a camera in the 3D view. Hence you can select the Spot light and switch to a view from its perspective by pressing Ctrl0 NumPad.

Spot Volumetric Effects



Spot lamps's Halo options

Spot lights also can produce “volumetric” effects. See Volumetric Light for more information about what it means.

Halo
: The Halo button allows a Spot lamp to have a volumetric effect applied to it. This button must be active if the volumetric effect is to be visible. Note that if you are using buffered shadows, you have extra options described in the Spot Buffered Shadows page.

Intensity
: The Intensity slider controls how intense/dense the volumetric effect is that is generated from the light source. The lower the value of the Intensity slider, the less visible the volumetric effect is, while higher Intensity values give a much more noticeable and dense volumetric effect.

Step
: This field can have a value between **0** and **12**. It is used to determine whether this Spot will cast volumetric shadows, and what quality those volumetric shadows will have.
If Step is set to a value of **0**, then no volumetric shadow will be generated.
Unlike most other controls, as the Step value increases, the quality of volumetric shadows decreases (but take less time to render), and *vice versa*.

Step values

A value of **8** for Halo Step is usually a good compromise between speed and accuracy.

Blender only simulates volumetric lighting in Spot lamps when using its internal renderer. This can lead to some strange results for certain combinations of settings for the light’s Energy and the halo's Intensity.
For example, having a Spot light with null or very low light Energy settings but a very high halo Intensity setting can result in a dark/black halo, which would not happen in the real world. Just be aware of this possibility when using halos with the internal renderer.

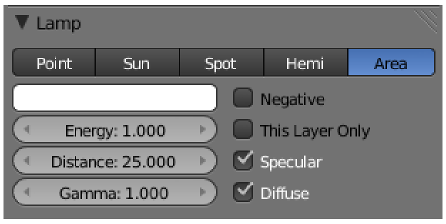
Note
The halo effect can be greatly enhanced when using buffered shadows: when the halo's Step is not null, they can create “volumetric shadows”. See the page about Spot Buffered Shadows for more information.

See Also

- Shadows
- Spot Lamp
- Spot Buffered Shadows

Area Lamp

The Area lamp simulates light originating from a surface (or surface-like) emitter. For example, a TV screen, your supermarket's neon lamps, a window, or a cloudy sky are just a few types of area lamp. The area lamp produces shadows with soft borders by sampling a lamp along a grid the size of which is defined by the user. This is in direct contrast to point-like artificial lights which produce sharp borders.



Energy: 1.000
Distance: 25.000
Gamma: 1.000

Commons Options

Lamp options

Distance, Energy and Color

These settings are common to most types of lamps, and are described in Light Properties.

Note that the Distance setting is much more sensitive and important for Area lamps than for others; usually any objects within the range of Distance will be blown out and overexposed. For best results, set the Distance to just below the distance to the object that you want to illuminate.

Gamma
: Amount to gamma correct the brightness of illumination. Higher values give more contrast and shorter falloff.

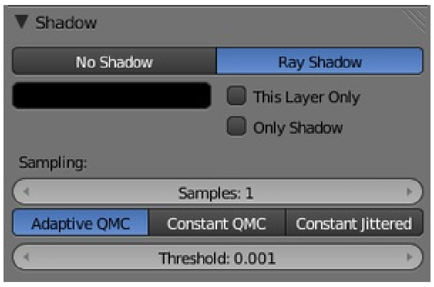
The Area lamp doesn't have light falloff settings. It uses an "inverse quadratic" attenuation law. The only way to control its falloff is to use the Distance and/or Gamma settings.

This Layer Only, Negative, Specular and Diffuse
: These settings control what the lamp affects, as described in What Light Affects.

Shadows

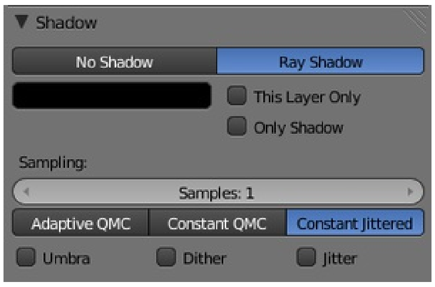
Area light ray-traced shadows are described here: Raytraced Shadows.

When an Area light source is selected, the Shadow panel has the following default layout:



Samples: 1
Threshold: 0.001

Adaptive QMC settings



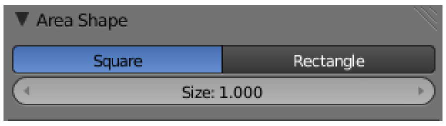
Samples: 1

Constant Jittered settings

The Shadow panel when Area light source is selected

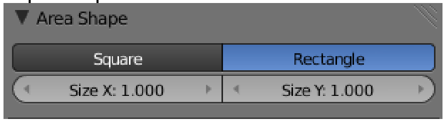
Area Shape

The shape of the area light can be set to Square or Rectangle.



Size: 1.000

Square options



Size X: 1.000
Size Y: 1.000

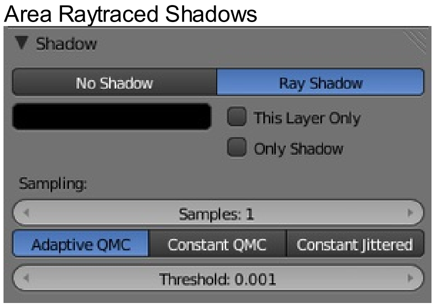
Rectangle options

Square/Rectangular
: Emit light from either a square or a rectangular area

Size/Size X/Size Y
: Dimensions for the Square or Rectangle

Shape Tips
Choosing the appropriate shape for your Area light will enhance the believability of your scene. For example, you may have an indoor scene and would like to simulate light entering through a window. You could place a Rectangular area lamp in a window (vertical) or from neons (horizontal) with proper ratios for Size X and Size Y. For the simulation of the light emitted by a TV screen a vertical Square area lamp would be better in most cases.

Area Raytraced Shadows



Adaptive QMC settings

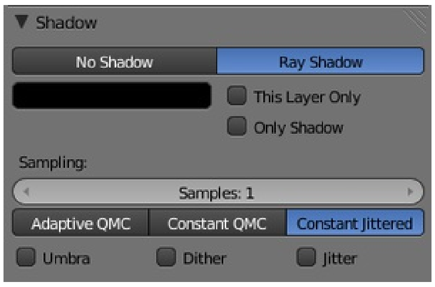
The Area light source can only cast ray-traced shadows. The ray-traced shadows settings of this lamp are mostly shared with other lamps, as described in Raytraced Properties. However, there are some specifics with this lamp, which are detailed below:

Shadow Samples

Samples
: This have the same role as with other lamps, but when using a Rectangle Area lamp, you have two samples settings: Samples X and Samples Y, for the two axes of the area plane.
Note also that when using the Constant Jittered sample generator method, this is more or less equivalent to the number of virtual lamps in the area. With QMC sample generator methods, it behaves similarly to with Lamp or Spot lamps.

Sample Generator Types

Adaptive QMC;Constant QMC
: These common setting are described in Shadow Properties.



Constant Jittered settings

Constant Jittered
: The Area lamp has a third sample generator method, Constant Jittered, which is more like simulating an array of lights. It has the same options as the old one: Umbra, Dither and Jitter.

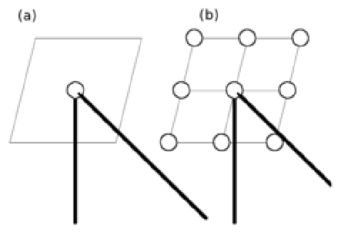
The following three parameters are only available when using the Constant Jittered sample generator method, and are intended to artificially boost the “soft” shadow effect, with possible loss in quality:

Umbra
: Emphasizes the intensity of shadows in the area fully within the shadow rays. The light transition between fully shadowed areas and fully lit areas changes more quickly (i.e. a sharp shadow gradient). You need Samples values equal to or greater than **2** to see any influence of this button.

Dither
: Applies a sampling over the borders of the shadows, similar to the way anti-aliasing is applied by the OSA button on the borders of an object. It artificially softens the borders of shadows; when Samples is set very low, you can expect poor results, so Dither is better used with medium Samples values. It is not useful at all with high Samples values, as the borders will already appear soft.

Jitter
: Adds noise to break up the edges of solid shadow samples, offsetting them from each other in a pseudo-random way. Once again, this option is not very useful when you use high Samples values where the drawback is that noise generates quite visible graininess.

Technical Details



Principles behind the Area light

The (*Principles behind the Area light*) picture helps to understand how the soft shadows are simulated.

`(a)` is the Area light as defined in Blender. If its shape is Square, then the softness of the shadow is defined by the number of light Samples in each direction of the shape. For example, `(b)` illustrates the equivalent case of an Area light (Square shape), with Samples set at **3** on the Shadow and Spot panel.

The Area lamp is then considered as a grid with a resolution of three in each direction, and with a light "dupliverted" at each node for a total of nine lights.

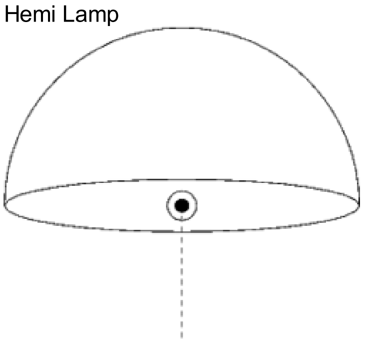
In case `(a)`, the energy (`E`) is `E/1`, and in case `(b)`, the energy of each individual pseudo-light is equal to `E/(Nbr of lights)`. Each pseudo-light produces a faint shadow (proportional to its energy), and the overlay of the shadows produces the soft shadow (it is darker where the individual shadows overlap, and lighter everywhere else).

Hints

You will note that changing the Size parameter of your area lamp doesn’t affect the lighting intensity of your scene. On the other hand, rescaling the lamp using the S in the 3D View could dramatically increase or decrease the lighting intensity of the scene. This behavior has been coded this way so that you can fine tune all your light settings and then decide to scale up (or down) the whole scene without suffering from a drastic change in the lighting intensity. If you only want to change the dimensions of your Area lamp, without messing with its lighting intensity, you are strongly encouraged to use the Size button(s) instead.

If your computer isn’t very fast, when using the Constant Jittered sample generator method, you could find it useful to set a low Samples value (like **2**) and activate Umbra, Dither, and/or Jitter in order to simulate slightly softer shadows. However, these results will never be better than the same lighting with high Samples values.

Hemi Lamp

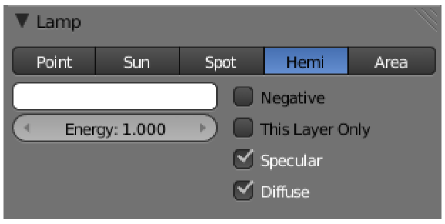

Hemi light conceptual scheme

The Hemi lamp provides light from the direction of a 180° hemisphere, designed to simulate the light coming from a heavily clouded or otherwise uniform sky. In other words, it is a light which is shed, uniformly, by a glowing dome surrounding the scene.

Similar to the Sun lamp, the Hemi’s location is unimportant, while its orientation is key.

The Hemi lamp is represented with four arcs, visualizing the orientation of the hemispherical dome, and a dashed line representing the direction in which the maximum energy is radiated, the inside of the hemisphere.

Options



Energy: 1.000

Negative

This Layer Only

✓ Specular

✓ Diffuse

Hemi lamp's panel

Energy and Color
: These settings are common to most types of lamps, and are described in Light Properties.

Layer, Negative,Specular, and Diffuse
: These settings control what the lamp affects, as described in What Light Affects.

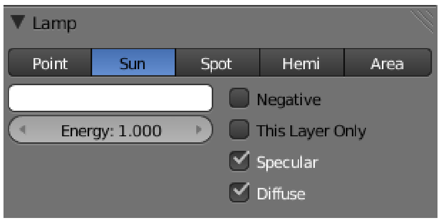
The Hemi lamp has no light falloff settings: it always uses a constant attenuation (i.e. no attenuation).

Since this lamp is the only lamp which cannot cast any shadow, the Shadow panel is absent.

Sun Lamp

A Sun lamp provides light of constant intensity emitted in a single direction. A Sun lamp can be very handy for a uniform clear daylight open-space illumination. In the 3D view, the Sun light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light.

This direction can be changed by rotating the Sun lamp, like any other object, but because the light is emitted in a constant direction, the location of a Sun lamp does not affect the rendered result (unless you use the “sky & atmosphere” option).


Lamp
Point [✓ Sun] Spot Hemi Area
Energy: 1.000
☐ Negative
☐ This Layer Only
✓ Specular
✓ Diffuse

Sun lamp panel

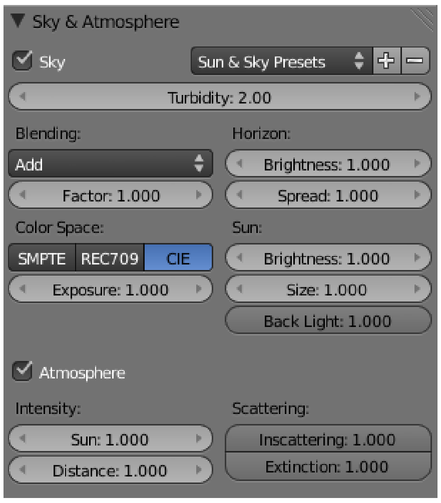
Lamp options

Energy and Color
: These settings are common to most types of lamps, and are described in Light Properties.

Negative, This Layer Only, Specular, and Diffuse
: These settings control what the lamp affects, as described in What Light Affects.

The Sun lamp has no light falloff settings: it always uses a constant attenuation (i.e. no attenuation!).

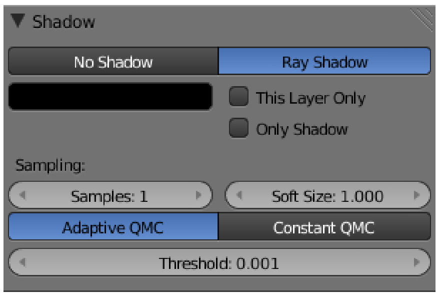
Sky & Atmosphere


Sky & Atmosphere
✓ Sky   Sun & Sky Presets
Turbidity: 2.00
Blending: Add
Factor: 1.000
Color Space: SMPTE REC709 [✓ CIE]
Exposure: 1.000
Horizon:
Brightness: 1.000
Spread: 1.000
Sun:
Brightness: 1.000
Size: 1.000
Back Light: 1.000
✓ Atmosphere
Intensity:
Sun: 1.000
Distance: 1.000
Scattering:
Inscattering: 1.000
Extinction: 1.000

Sky & Atmosphere panel

Various settings for the appearance of the sun in the sky, and the atmosphere through which it shines, are available. For details, see Sky and Atmosphere.

Shadow

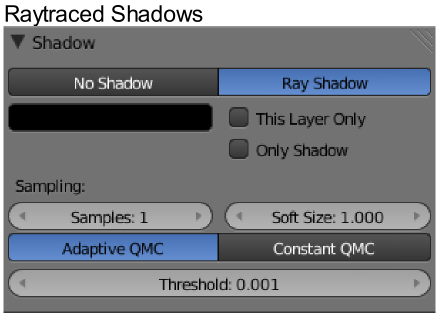

Shadow
No Shadow [✓ Ray Shadow]
☐ This Layer Only
☐ Only Shadow
Sampling:
Samples: 1
Soft Size: 1.000
[✓ Adaptive QMC] Constant QMC
Threshold: 0.001

Shadow panel

The Sun light source can only cast ray-traced shadows. It shares with other lamp types the same common shadowing options, described in Shadows Properties.

The ray-traced shadows settings of this lamp are shared with other lamps, and are described in Raytraced Properties.

Raytraced Shadows

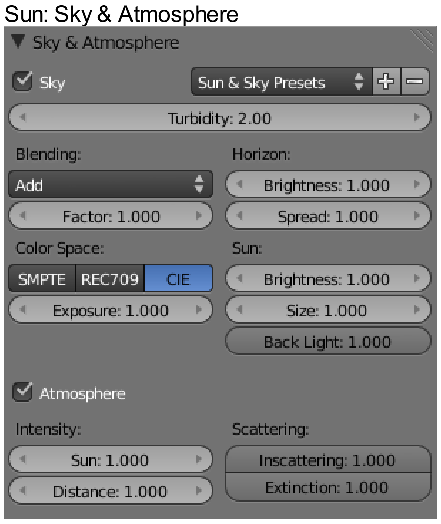


Shadow panel

The Sun light source can only cast raytraced shadows. It shares with other lamp types the same common shadowing options, described in Shadows Properties.

The raytraced shadows settings of this lamp are shared with other ones, and are described in Raytraced Properties.

Sun: Sky & Atmosphere

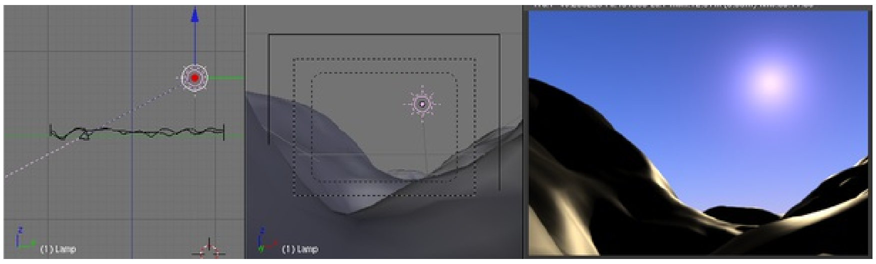


Sky & Atmosphere panel

This panel allows you to enable an effect that simulates various properties of real sky and atmosphere: the scattering of sunlight as it crosses the kilometers of air overhead. For example, when the Sun is high, the sky is blue (and the horizon, somewhat whitish). When the Sun is near the horizon, the sky is dark blue/purple, and the horizon turns orange. The dispersion of the atmosphere is also more visible when it is a bit foggy: the farther away an object is, the more “faded” in light gray it is… Go out into the countryside on a nice hot day, and you will see.

To enable this effect, you have to use a Sun light source. If, as usual, the *position* of the lamp has no importance, its *rotation* is crucial: it determines which hour it is. As a starting point, you should reset rotation of your Sun (with AltR, or typing **0** in each of the three Rotation fields X/Y/Z in the Transform Properties panel – N). This way, you’ll have a nice mid-day sun (in the tropics).

Now, there are two important angles for the Sky/Atmosphere effect: the “incidence” angle (between the light direction and the X-Y plane), which determines the “hour” of the day (as you might expect, the default rotation – straight down – is “mid-day”, a light pointing straight up is “midnight”, and so on…). And the rotation around the Z axis determines the position of the sun around the camera.



The dashed “light line” of the Sun lamp crossing the camera focal point.

In fact, to have a good idea of where the sun is in your world, relative to the camera in your 3D view, you should always try to have the dashed “light line” of the lamp crossing the center of the camera (its “focal” point), as shown in (*The dashed “light line” of the Sun lamp crossing the camera focal point*). This way, in camera view (0 NumPad, center window in the example picture), you will see where the “virtual” sun created by this effect will be.

It is important to understand that the *position* of the sun has no importance for the effect: only its *orientation* is relevant. The position just might help you in your scene design.

Options

Sun & Sky Presets

- Classic:
- Desert:
- Mountain:

Sky

Sky
: This button enables the sky settings: it will create a “sky”, with a “sun” if visible, and mix it with the background as defined in World settings.

Turbidity
: This is a general parameter that affects sun view, sky and atmosphere; it’s an atmosphere parameter where low values describe clear sky, and high values shows more foggy sky. In general, low values give a clear, deep blue sky, with “little” sun; high values give a more reddish sky, with a big halo around the sun. Note that this parameter is one which can really modify the “intensity” of the sun lighting. See examples below.

Here are its specific controls:

Blending

- The first drop-down list shows you a menu of various mix methods. The one selected will be used to blend the sky and sun with the background defined in the World settings. The mixing methods are the same as described e.g. in the Mix Compositing Node page.
- Factor

  Controls how much the sky and sun effect is applied to the World background.

Color space
: These buttons allows you to select which color space the effect uses, with the following choices:

  - CIE
  - REC709
  - SMPTE
  - Exposure

    This numeric field allows you to modify the exposure of the rendered Sky and Sun (**0.0** for no correction).

Horizon

- Brightness

  Controls brightness of colors at the horizon. Its value should be in the range **0.0** to **10.0**; values near zero means no horizontal brightness, and large values for this parameter increase horizon brightness. See examples below.

- Spread

  Controls spread of light at the horizon. Its value should be in the range **0.0** to **10.0**; values low in the range result in less spread of light at horizon, and values high in the range result in horizon light spread in through all the sky.

Sun

- Brightness

  Controls the sun brightness. Its value should be in the range **0.0** to **10.0**; with low values the sky has no sun and with high values the sky only has sun.

- Size

  Controls the size of sun. Its values should be in the range **0.0** to **10.0**, but note that low values result in large sun size, and high values result in small sun size. Note that the overall brightness of the sun remains constant (set by Brightness), so the larger the sun (the smaller Size), the more it “vanishes” in the sky, and *vice versa*.

- Back Light

  For “Back Scatter Light”, result on sun’s color, high values result in more light around the sun. Its values range is **-1.0** to **1.0**. Negative values result in less light around sun.

Atmosphere

Atmosphere
: This button enables the atmosphere settings. It will not modify the background, but it tries to simulate the effects of an atmosphere: scattering of the sunlight in the atmosphere, its attenuation, …

Intensity

- Sun

  Sets sun intensity. Its values are in range **0.0** to **10.0**. High values result in bluer light on far objects.

- Distance

  This factor is used to convert Blender units into an understandable unit for atmosphere effect, it starts from **0** and high values result in more yellow light in the scene.

Scattering

- Inscattering

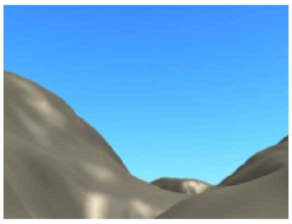
  This factor can be used to decrease the effect of light inscattered into atmosphere between the camera and objects in the scene. This value should be **1.0** but can be changed to create some nice, but not realistic, images.

- Extinction

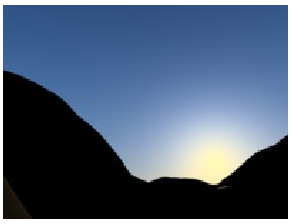
  This factor can be use to decrease the effect of extinction light from objects in the scene. Like Inscattering factor, this parameter should be **1.0** but you can change it; low values result in less light extinction. Its value is in the range **0.0** to **1.0**.

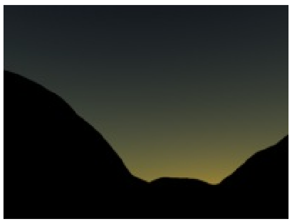
Examples

First, let's see what happens when we modify the orientation of the sun:


With sun right overhead (mid-day).

With sun deep "under the Earth" (midnight).

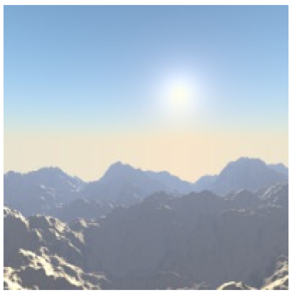

Sun slightly above the horizon (start of twilight).

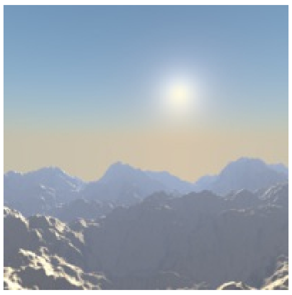

Sun slightly below the horizon (end of twilight).

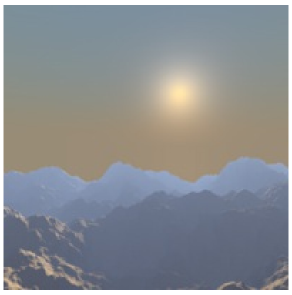
Variations in Sun orientation, Sun Size to **5.0**, all other settings to default.

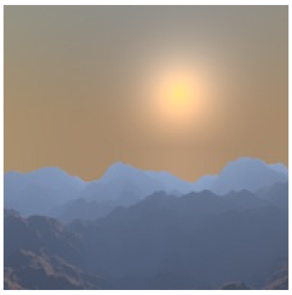
The 2.4 .blend file of these examples.

And now, the effects of various settings (examples created with this 2.4 .blend file):


Turbidity: **2.0**.

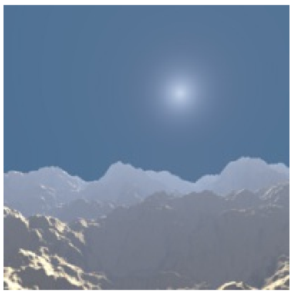

Turbidity: **2.3**.

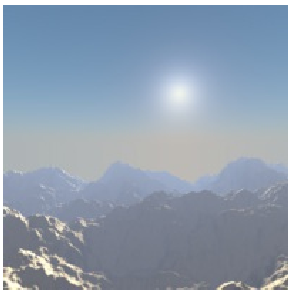

Turbidity: **5.0**.

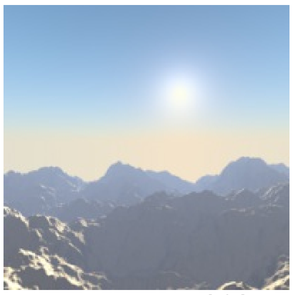

Turbidity: **10.0**.

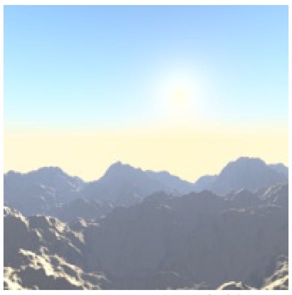
Variations in Turbidity parameter, all other settings to default.

Sky

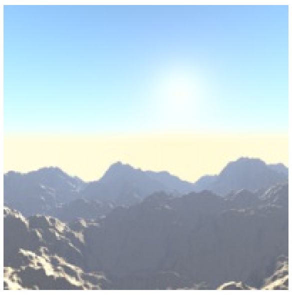

Horizon Brightness: **0.0**.

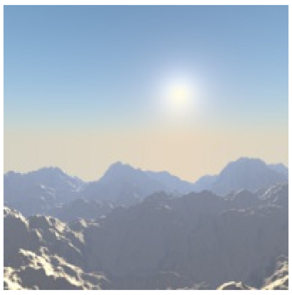

Horizon Brightness: **0.85**.

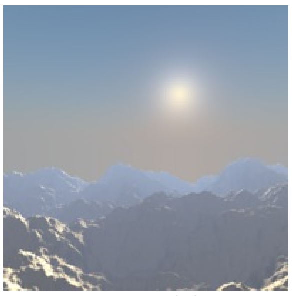

Horizon Brightness: **1.04**.

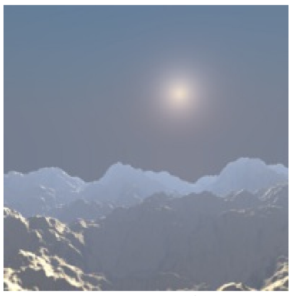

Horizon Brightness: **1.13**.

Variations in Horizon Brightness parameter, all other settings to default.

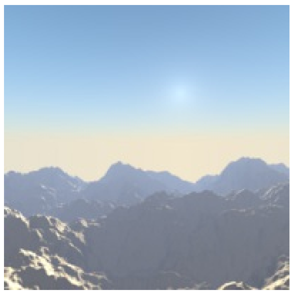

Horizon Spread: **0.7**.

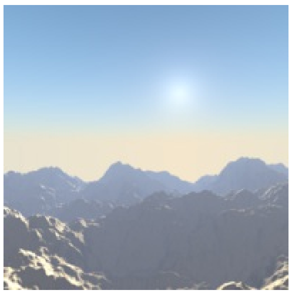

Horizon Spread: **1.2**.

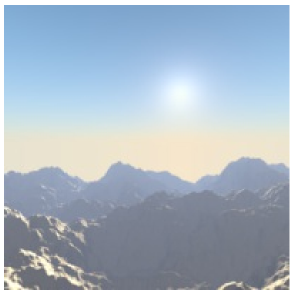

Horizon Spread: **2.2**.

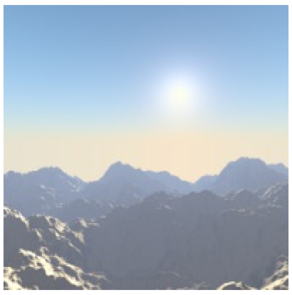

Horizon Spread: **5.0**.

Variations in Horizon Spread parameter, all other settings to default.

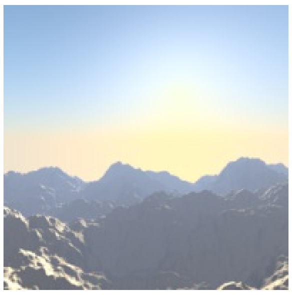

Sun Brightness: **0.2**.

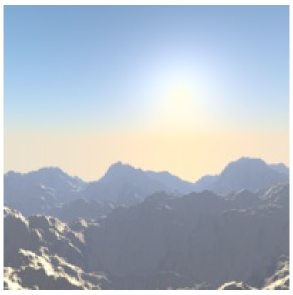

Sun Brightness: **0.5**.


Sun Brightness: **0.75**.


Sun Brightness: **1.0**.

Variations in Sun Brightness parameter, all other settings to default.


Sun Size: **2.0**.


Sun Size: **4.0**.

Sun Size: **7.0**.

Sun Size: **10.0**.

Variations in Sun Size parameter, all other settings to default.

Back Light: **-1.0**.

Back Light: **-0.33**.

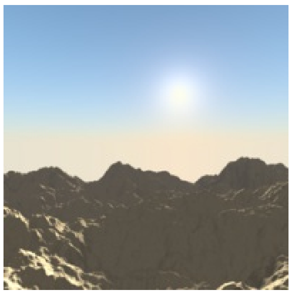
Back Light: **0.33**.

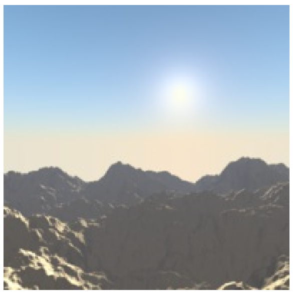
Back Light: **1.0**.

Variations in Back Light parameter, Sun Bright to **2.5**, all other settings to default.

Atmosphere

For all renders below, Hor.Bright is set to **0.2**, and Sun Bright to **2.0**.

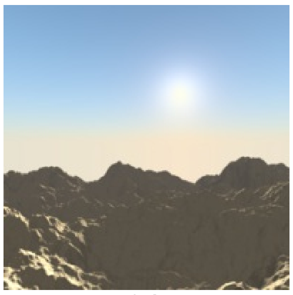

Sun Intensity: **1.0**.

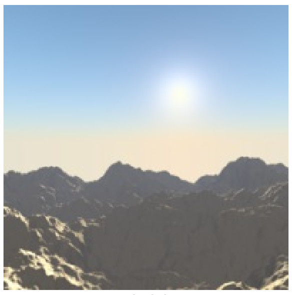

Sun Intensity: **3.33**.

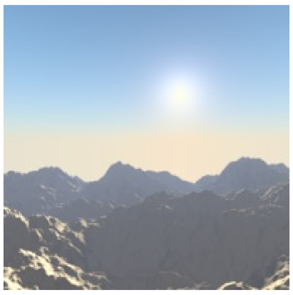
Sun Intensity: **6.66**.

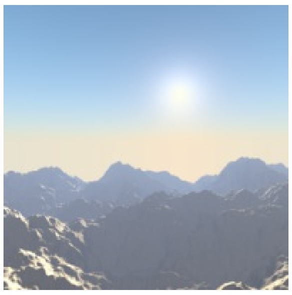
Sun Intensity: **10.0**.

Variations in Sun Intensity parameter, all other settings to default.

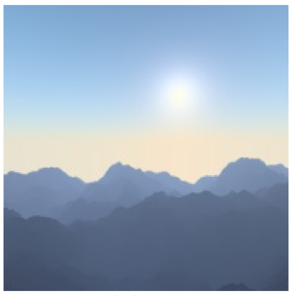

Inscattering: **0.1**.

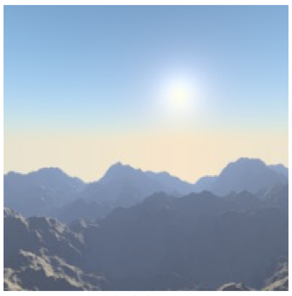

Inscattering: **0.33**.

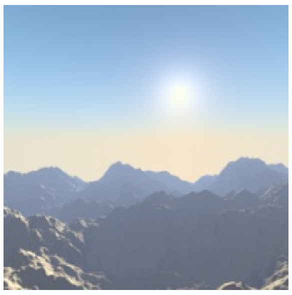

Inscattering: **0.66**.

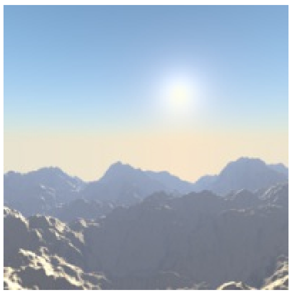

Inscattering: **1.0**.

Variations in Inscattering parameter, all other settings to default.

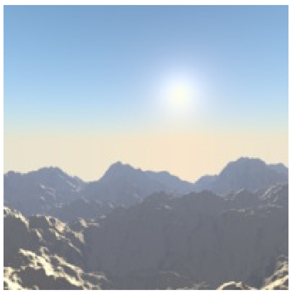

Extinction: **0.0**.

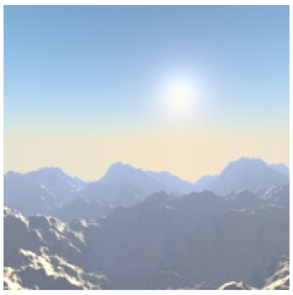

Extinction: **0.33**.

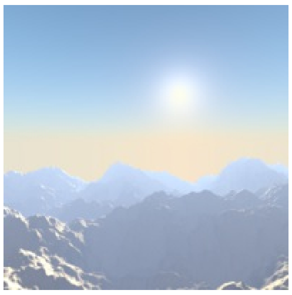

Extinction: **0.66**.

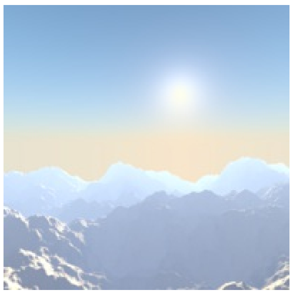

Extinction: **1.0**.

Variations in Extinction parameter, all other settings to default.


Distance: **1.0**.


Distance: **2.0**.


Distance: **3.0**.


Distance: **4.0**.

Variations in Distance parameter, all other settings to default.

Hints and limitations

To always have the Sun pointing at the camera center, you can use a TrackTo constraint on the sun object, with the camera as target, and -Z as the “To” axis (use either X or Y as “Up” axis). This way, to modify height/position of the sun in the rendered picture, you just have to move it; orientation is automatically handled by the constraint. Of course, if your camera itself is moving, you should also add e.g. a Copy Location constraint to your Sun lamp, with the camera as target – and the Offset option activated… This way, the sun light won’t change as the camera moves around.

If you use the default Add mixing type, you should use a very dark-blue world color, to get correct “nights”…

This effect works quite well with a Hemi lamp, or some ambient occlusion, to fill in the Sun shadows.

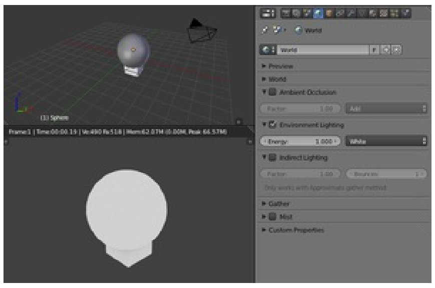
Atmosphere shading currently works incorrectly in reflections and refractions and is only supported for solid shaded surfaces. This will be addressed in a later release.

Lighting Rigs

A rig is a standard setup and combination of objects; there can be lighting rigs, or armature rigs, etc. A rig provides a basic setup and allows you to start from a known point and go from there. Different rigs are used for different purposes and emulate different conditions; the rig you start with depends on what you want to convey in your scene. Lighting can be very confusing, and the defaults do not give good results. Further, very small changes can have a dramatic effect on the mood and colors. At major studios, lighting is an entire step and specialty. Well, let’s get out of the darkness of confusion and let me en*light*en you.

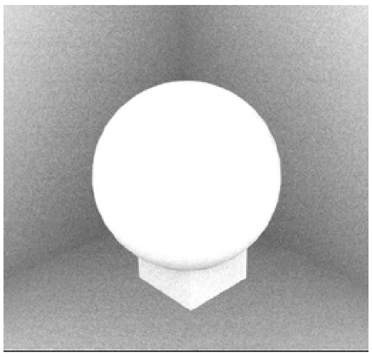
In all the lighting rigs, the default camera is always positioned nearly **15** degrees off dead-on, about **25 BU** (Blender Units) back and **9 BU** to the side of the subject, at eye level, and uses a long lens (**80mm**). Up close, a **35mm** lens will distort the image. A long lens takes in more of the scene. A dead-on camera angle is too dramatic and frames too wide a scene to take in. So now you know; next time you go to a play, sit off-center and you won’t miss the action happening on the sidelines and will have a greater appreciation for the depth of the set. Anyway, enough about camera angles; this is about lighting.

Environment or Ambient Only



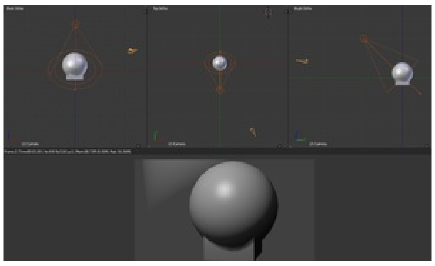
Environment (Ambient) lighting only.

In the World context, there is a panel Environment Lighting, where you enable environment or ambient lighting of your scene. Ambient light is the scattered light that comes from sunlight being reflected off every surface it hits, hitting your object, and traveling to camera.


Ambient occlusion.

Ambient light illuminates, in a perfectly balanced, shadeless way, without casting shadows. You can vary the intensity of the ambient light across your scene via ambient occlusion. The ambient color is a sunny white.

Single Rig


Standard Spot light rig.

The sole, or key, spot light rig provides a dramatic, showy, yet effective illumination of one object or a few objects close together. It is a single Spot light, usually with a hard edge. Halos are enabled in this render to remind you of a smoky nightclub scene. It is placed above and directly in front of the subject; in this case **10 BU** in front and **10 BU** high, just like a stage, it shines down at about a **40** degrees angle. We use quadratic attenuation.

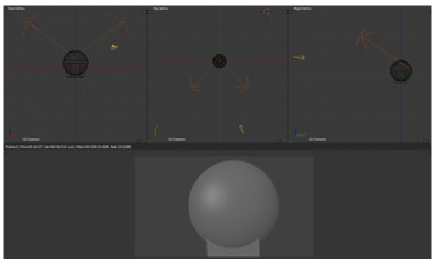
You can make the spot wider by increasing Size Spot Shape and softening the edge by increasing Blend Spot Shape, and parent it to the main actor, so that the spot follows him as he moves around. Objects close to the main actor will naturally be more lit and your viewer will pay attention to them.

Moving this spot directly overhead and pointing down gives the interrogation effect. At the opposite end of the show-off emotional spectrum is one soft candlelight (Point lamp, short falloff Distance, yellow light) placed really up close to the subject, dramatizing the fearful “lost in the darkness” effect.

Somewhere in the macabre spectrum is a hard spot on the floor shining upward. For fun, grab a flashlight, head into the bathroom and close the door. Turn out the light and hold the flashlight under your chin, pointing up. Look in the mirror and turn it on. Ghoulies! Don’t blame me for nightmares, and I hope you get the point: lighting, **even with a single light, varying the intensity, location and direction, changes everything** in a scene.

Use this rig, with Environment Lighting light (and props receiving and being lit by ambient light in their material settings) for scenes that feature one main actor or a product being spotlighted. Do not use this rig for big open spaces or to show all aspects of a model.

Two-Point Rig


Standard two-point light rig.

The two-point lighting rig provides a balanced illumination of an object. Shown to the right are the views of the standard two-point lighting rig. It is called the two-point because there are two points of light. The standard two-point lighting rig provides a balanced illumination of untextured objects hanging out there in 3D space. This rig is used in real studios for lighting a product, especially a glossy one.

Both lights are almost the same but do different things. Both emulate very wide, soft light by being Hemi. In real life, these lights bounce light off the inside of a silver umbrella.

Notice how we use low Energy to bring out the dimensionality of the sphere; I can't stress that enough. Hard, bright lights actually flatten it and make you squint. Soft lights allow your eye to focus. We disable specular for right Hemi, so we don't get that shiny forehead or nose.

The lamp on the left however, lets it be known that it is there by enabling specular; specular flare is that bright spot that is off center above midline on the sphere.

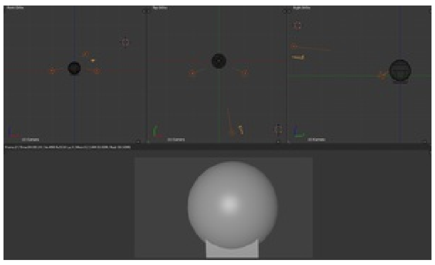
Use this rig to give even illumination of a scene, where there is no main focus. The Hemi's will light up background objects and props, so Environment Lighting is not that important. At the opposite end of the lighting spectrum, two narrow Spot lights at higher power with a hard edge gives a "This is the Police, come out with your hands up" kind of look, as if the subject is caught in the crossfire.

Three-Point Rigs

The standard three-point lighting rig is the most common illumination of objects and scenes bar none. If you want to show off your model, use this rig. As you can see, the untextured unmaterialized sphere seems to come out at you. There are multiple thesis on this rig, and you will use one of two:

- Studio – used in a real studio to film in front of a green screen or backdrop. Use this rig when you are rendering your CG objects to alpha into the scene so that the lighting on the actors *and* your CG objects is the same.
- Standard – used in real life to light actors on a set, and gives some backlighting to highlight the sides of actors, making them stand out more and giving them depth.

Studio rig


Studio three-point light rig.

Shown to the right are the "Studio" top, front, and side views of the standard three-point lighting rig. It changes the dynamics of the scene, by making a brighter "key" light give some highlights to the object, while two side "fill" lights soften the shadows created by the key light.

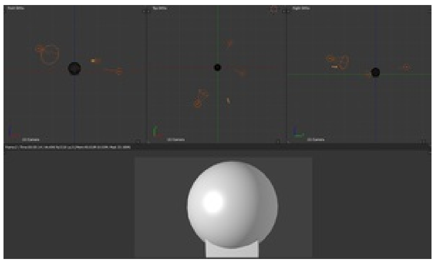
In the studio, use this rig to film a talking head (actor) in front of a green screen, or with multiple people, keeping the key light on the main actor. This rig is also used to light products from all angles, and the side fill lights light up the props.

The key light is the Area light placed slightly above and to the left of the camera. It allows the specular to come out. It is about **30 BU** back from the subject, and travels with the camera. A little specular shine lets you know there's a light there, and that you're not looking at a ghost. In real life, it is a spot with baffles, or blinders, that limit the area of the light.

The two sidelights are reduced to only fill; each of them are Hemi lights placed **20 BU** to the side and **5 BU** in front of the subject, at ground level. They don't cause a spotshine on the surface by disabling specular, and at ground level, light under the chin or any horizontal surfaces, countering the shadows caused by the key light.

Use this rig to give balanced soft lighting that also highlights your main actor or object. It combines the best of both the single rig and the two-point rig, providing balanced illumination and frontal highlights. For a wide scene, you may have to pull the sidelights back to be more positioned like the two-point rig.

Standard Rig

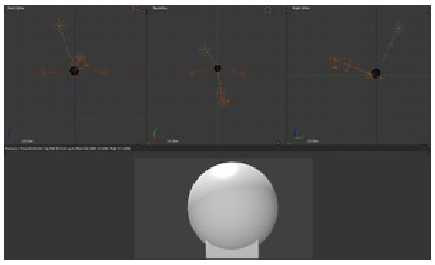

Standard three-point light rig.

Without a curtain in back of your main subject, you have depth to work with. The left fill light has been moved behind the subject (so it is now called a backlight) and is just off-camera, while the right side fill light remains the same. The keylight gives you specular reflection so you can play with specularity and hardness in your object’s material settings. The key light gives that “in-the-spotlight” feel, highlighting the subject, while the backlight gives a crisp edge to the subject against the background. This helps them stand out.

In this rig, the key light is a fairly bright spot light. Use a slighter tinge of yellow because the light is so bright; it is the only light for that side. The other sidelight has been moved in back and raised to eye (camera) level. You need to cut the energy of the backlight in half, or when it is added to the remaining sidelight, it will light up the side too much and call too much attention to itself. You can vary the angle and height of the backlight to mimic a sun lighting up the objects.

Use this rig in normal 3D animations to light the main actor. Use this rig especially if you have transparent objects (like glass) so that there is plenty of light to shine through them to the camera. The tricky part here is balancing the intensities of the lights so that no one light competes with or overpowers the others, while making sure all three work together as a team.

Four-point Rig


Four-point light rig.

The four-point lighting rig provides a better simulation of outside lighting, by adding a Sun lamp **30** Blender Units above, **10** to the side, and **15 BU** behind the subject. This sunlight provides backlighting and fills the top of the subject; even producing an intentional glare on the top of their head, telling you there is a sun up there. Notice it is colored yellow, which balances out the blue sidelights.

Changing the key light to a Spot, select Inverse Square, disable Specular and pure white light combines with and softens the top sun flare while illuminating the face, resulting in a bright sunshine effect. Two lights above means sharper shadows as well, so you might want to adjust the side fill lights. In this picture, they are still Hemi, disable Specular.

Use this rig when the camera will be filming from behind the characters, looking over their shoulder or whatnot, because the sun provides the backlight there. Also use this rig when you have transparent objects, so there is light to come through the objects to the camera.

Another spot for the fill light is shining up onto the main actor’s face, illuminating the underside of his chin and neck. This gets rid of a sometimes ugly shadow under the chin, which if not corrected, can make the actor look fat or like they have a double chin; otherwise distracting. It evens out the lighting of the face.

Troubleshooting

If you run into a problem with your render, where there are really bright areas, or really dark ones, or strange shadows, or lines on your objects, here is what I suggest you do:

1. First, try deactivating all materials (create a default, gray one, and enter its name in the Mat field, Layer panel, Render Layers context – to get back all your normal materials, just erase this text field!). See if you get those problems with just grayness objects. If you don’t have the problem anymore, that should tell you that you’ve got a materials-interacting-with-light problem. Check the material settings, especially ambient, reflection and all those little buttons and sliders in the Material context . You can set some lights to affect only certain materials, so if there’s an issue with only a few objects being really bright, start with those.
2. Then start “killing” lights (e.g. moving them to an unused layer); regress all the way back to one light, make sure it’s smooth, then add them in one by one. As they add together, reduce power in the tested ones so they merge cleanly, or consider not adding it at all, or, especially, reduce the energy of the lamp you just introduced.
3. You can also set lights to only light objects on a layer, so again, if some of the gray spheres have weirdness, check for that as well. Again, you may have done some of this accidentally, so sometimes deleting the light and re-adding it with defaults helps

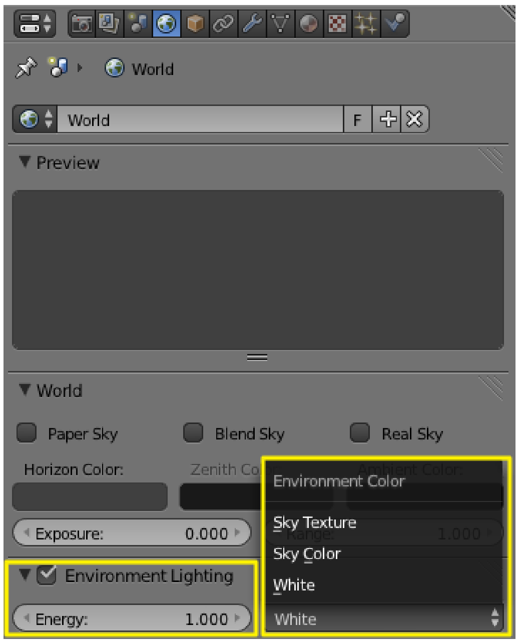
you reset to a known-good situation.

4. Negative lights can be very tricky, and make your model blotchy, so pay special attention to your use of those special lights. Shadow-only lights can throw off the look of the scene as well. Overly textured lights can make your scene have random weird colors. Don’t go too far off a slight tinge of blue or yellow or shades of white, or your material may show blue in the Material context but render green, and you will be very confused.
5. Look at your environment settings World context: Horizon, Zenith, and Environment Lighting.

Environment Lighting

Environment light provides light coming from all directions.

Light is calculated with a ray-traced method which is the same as that used by Ambient Occlusion. The difference is that Environment lighting takes into account the "ambient" parameter of the material shading settings, which indicates the amount of ambient light/color that that material receives.



Environment Lighting panel.

Also, you can choose the environment color source (white, sky color, sky texture) and the light energy.

Energy
: Defines the strength of environment light.

Environment Color
: Defines where the color of the environment light comes from.

Using both settings simultaneously produces better global lighting.

It's good for mimicking the sky in outdoor lighting. Environment lighting can be fairly noisy at times.

Ambient Occlusion

Ambient Occlusion is a sophisticated ray-tracing calculation which simulates soft global illumination shadows by faking darkness perceived in corners and at mesh intersections, creases, and cracks, where ambient light is occluded, or blocked.

There is no such thing as AO in real life; AO is a specific not-physically-accurate (but generally nice-looking) rendering trick. It basically samples a hemisphere around each point on the face, sees what proportion of that hemisphere is occluded by other geometry, and shades the pixel accordingly.

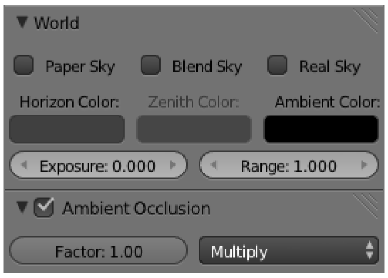
It's got nothing to do with light at all; it's purely a rendering trick that tends to look nice because generally in real life surfaces that are close together (like small cracks) will be darker than surfaces that don't have anything in front of them, because of shadows, dirt, etc.

The AO process, though, approximates this result; it's not simulating light bouncing around or going through things. That's why AO still works when you don't have any lights in the scene, and it's why just switching on AO alone is a very bad way of "lighting" a scene.

You must have ray tracing enabled as a Render panel option in the Shading section for this to work.

You must have an ambient light color set as you desire. By default, the ambient light color (world) is black, simulating midnight in the basement during a power outage. Applying that color as ambient will actually darken all colors. A good outdoor mid-day color is RGB (**0.9**, **0.9**, **0.8**) which is a whitish yellow sunny kind of color on a bright-but-not-harshly-bright day.

Options



The World panel with ambient color sliders highlighted.

Factor
: The strength of the AO effect, a multiplier for addition.

Ambient Occlusion is composited during the render. Two blending modes are available:

Add
: The pixel receives light according to the number of non-obstructed rays. The scene is lighter. This simulates global illumination.

Multiply
: Ambient occlusion is multiplied over the shading, making things darker.

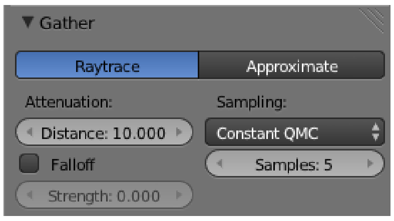
Note
If Multiply is chosen, there must be other light sources; otherwise the scene will be pitch black. In the other two cases the scene is lit even if no explicit light is present, just from the AO effect. Although many people like to use AO alone as a quick shortcut to light a scene, the results it gives will be muted and flat, like an overcast day. In most cases, it is best to light a scene properly with Blender's standard lamps, then use AO on top of that, set to "Multiply", for the additional details and contact shadows.

The Gather panel contains settings for the ambient occlusion quality. Note that these settings also apply to Environment Lighting and Indirect Lighting.

Ambient occlusion has two main methods of calculation: Raytrace and Approximate.

Gather

Raytrace



The Amb Occ panel, Raytrace method.

The Raytrace method gives the more accurate, but also the more noisy results. You can get a nearly noiseless image, but at the cost of render time… It is the only option if you want to use the colors of your sky's texture.

Attenuation

Length of rays defines how far away other faces may be and still have an occlusion effect. The longer this distance, the greater impact that far-away geometry will have on the occlusion effect. A high Distance value also means that the renderer has to search a greater area for geometry that occludes, so render time can be optimized by making this distance as short as possible for the visual effect that you want.

Sampling

Samples

The number of rays used to detect if an object is occluded. Higher numbers of samples give smoother and more accurate results, at the expense of slower render times. The default value of **5** is usually good for previews. The actual number of rays shot out is the square of this number (i.e. Samples at **5** means **25** rays). Rays are shot at the hemisphere according to a random pattern (determined by the sample methods described above); this causes differences in the occlusion pattern of neighboring pixels unless the number of shot rays is big enough to produce good statistical data.

Ambient Occlusion with **3** Samples.

Ambient Occlusion with **6** Samples.

Ambient Occlusion with **12** Samples.

You have the three standard sampling options:

Constant QMC

The base Quasi-Monte Carlo, gives evenly and randomly distributed rays.

Adaptive QMC

An improved version of QMC, that tries to determine when the sample rate can be lowered or the sample skipped, based on its two settings:

Threshold

The limit below which the sample is considered fully occluded (“black”) or un-occluded (“white”), and skipped.

Adapt to Speed

A factor to reduce AO sampling on fast-moving pixels. As it uses the Vec render pass, that must also be enabled (see render passes page).

About QMC

See also the raytraced shadows page for more info about the Quasi-Monte Carlo sampling method.

Constant Jittered

The historical sample method, more prone to “bias” artifacts…

Bias

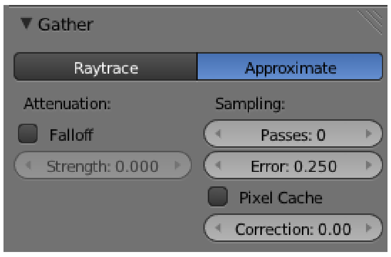
The angle (in radians) the hemisphere will be made narrower (i.e. the hemisphere will no longer be a real hemisphere: its section will no longer be a semicircle, but an arc of a circle of “`pi - Bias`” radians).

The bias setting allows you to control how smooth “smooth” faces will appear in AO rendering. Since AO occurs on the original faceted mesh, it is possible that the AO light makes faces visible even on objects with “smooth” on. This is due to the way AO rays are shot, and can be controlled with the Bias slider. Note that while it might even happen with QMC sampling methods, it is much more visible with the Constant Jittered one – and anyway, you have no Bias option for QMC.

24×24 UV Sphere with Bias: **0.05** (default). Note the

Raising the Bias to **0.15** removes the faceted facets on the sphere's surface even though it is set to smooth. artifacts.

Approximate



The Amb Occ panel, Approximate method.

The Approximate method gives a much smoother result for the same amount of render time, but as its name states, it is only an approximation of the Raytrace method, which implies it might produce some artifacts – and it cannot use the sky's texture as the base color

This method seems to tend to "over-occlude" the results. You have two complementary options to reduce this problem:

Passes
: Set the number of pre-processing passes, between **0** (no pre-processing) to **10**. Keeping the pre-processing passes high will increase render time but will also clear some artifacts and over-occlusions.

Error
: This is the tolerance factor for approximation error (i.e. the max allowed difference between approximated result and fully computed result). The lower, the slower the render, but the more accurate the results… Ranges between **0.0** and **10.0**, defaults to **0.250**.

Pixel Cache
: When enabled, it will keep values of computed pixels to interpolate it with its neighbors. This further speeds up the render, generally without visible loss in quality…

Correction
: A correction factor to reduce over-occlusion. Ranges between **0.0** (no correction) to **1.0**.

Common Settings

Falloff
: When activated, the distance to the occluding objects will influence the "depth" of the shadow. This means that the further away the occluding geometry is, the lighter its "shadow" will be. This effect only occurs when the Strength factor is higher than **0.0**. It mimics light dispersion in the atmosphere…

  Strength
  : Controls the attenuation of the shadows enabled with Use Falloff. Higher values give a shorter shadow, as it falls off more quickly (corresponding to a more foggy/dusty atmosphere). Ranges from **0.0** (default, no falloff) to **10.0**.

Technical Details

Ambient occlusion is calculated by casting rays from each visible point, and by counting how many of them actually reach the sky, and how many, on the other hand, are obstructed by objects.

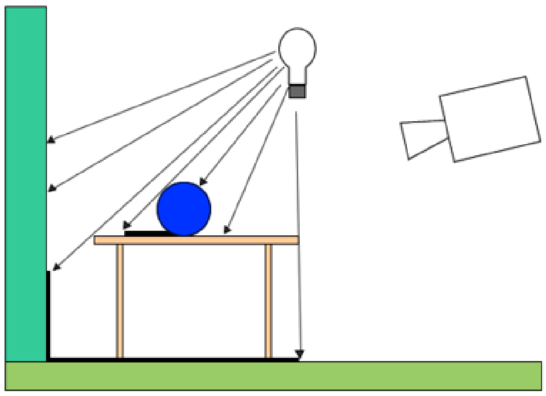
The amount of light on the point is then proportional to the number of rays which have "escaped" and have reached the sky. This is done by firing a hemisphere of shadow rays around. If a ray hits another face (it is occluded) then that ray is considered "shadow", otherwise it is considered "light". The ratio between "shadow" and "light" rays defines how bright a given pixel is.

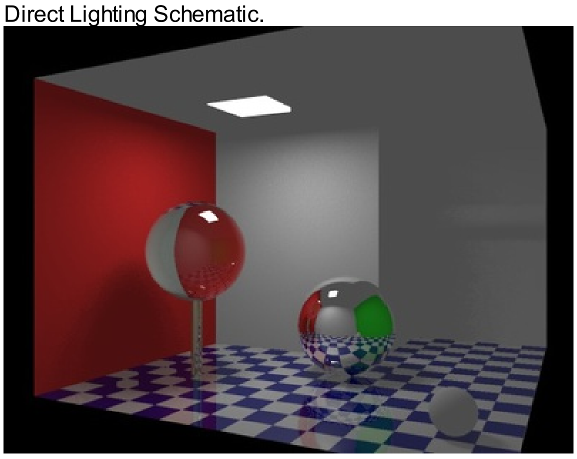
Hints

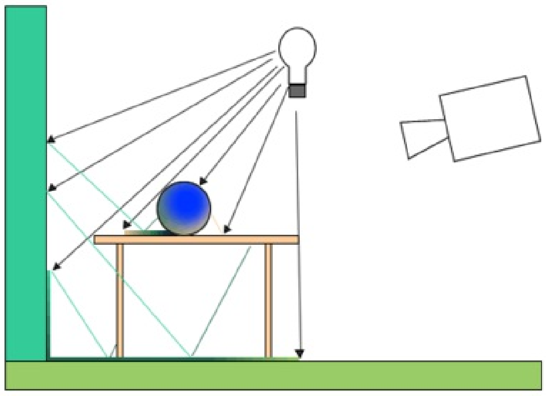
Ambient occlusion is a ray-tracing technique (at least with the Raytrace method), so it tends to be slow. Furthermore, performance severely depends on octree size, see the rendering chapter for more information.

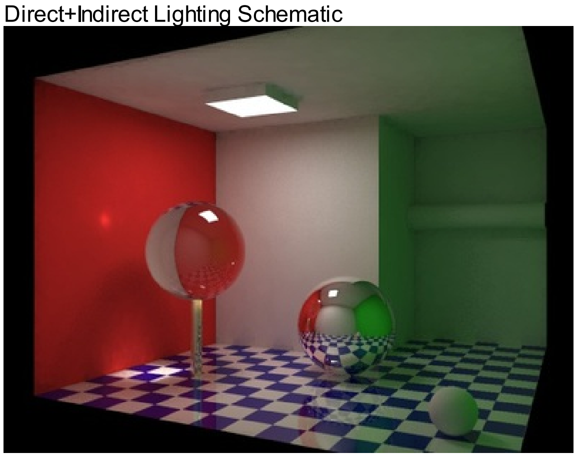
Inderect Lighting

Inderect Lighting adds inderect light bouncing of surrounding objects. It's modes the light that is reflected from other surfaces to the current surface. Is more comprehensive, more physically correct, and produces more realistic images. It is also more computationally expensive. Take a look at the following examples of a scene lit with Direct Lighting and both Direct+Indirect Lighting:


Direct Lighting Schematic.

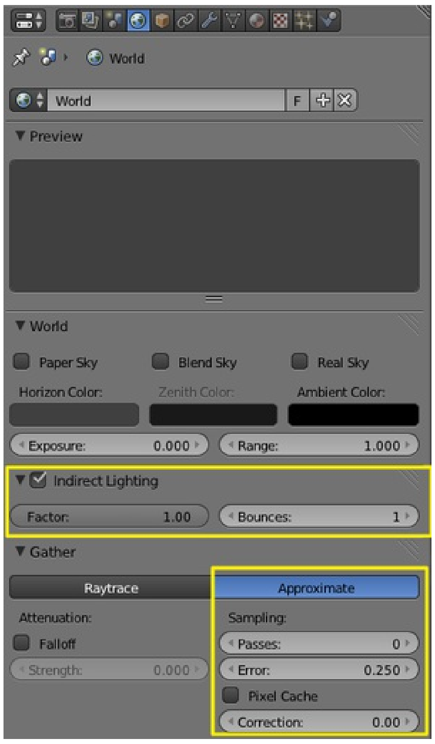

Direct Lighting Render


Direct+Indirect Lighting Schematic


Direct+Indirect Lighting Render

[Images courtesy]

Inderect Lighting only works with Approximate gather method.


World
World
Preview
World
Paper Sky
Blend Sky
Real Sky
Horizon Color:
Zenith Color:
Ambient Color:
Exposure: 0.000
Range: 1.000
✓ Indirect Lighting
Factor: 1.00
Bounces: 1
Gather
Raytrace
Approximate
Attenuation:
Falloff
Strength: 0.000
Sampling:
Passes: 0
Error: 0.250
Pixel Cache
Correction: 0.00

Inderect Lighting parameters.

Options

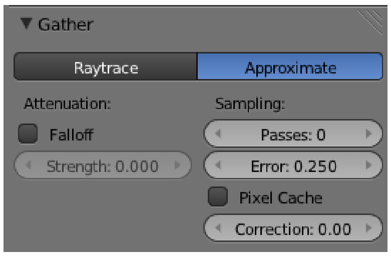
The Inderect Lighting panel contains two options:

Factor
: Defines how much surrounding objects contribute to light.

Bounces
: Number of inderect deffuse light bounces.

The Gather panel contains settings for the inderect lighting quality. Note that these settings also apply to Environment Lighting and Ambient Occlusion.

Approximate


Gather
Raytrace
Approximate
Attenuation:
Falloff
Strength: 0.000
Sampling:
Passes: 0
Error: 0.250
Pixel Cache
Correction: 0.00

The Inderect Lighting panel,
Approximate method.

The Approximate method gives a much smoother result for the same amount of render time, but as its name states, it is only an approximation of the Raytrace method, which implies it might produce some artifacts – and it cannot use the sky's texture as the base color

This method seems to tend to "over-occlude" the results. You have two complementary options to reduce this problem:

Passes
: Set the number of pre-processing passes, between **0** (no pre-processing) to **10**. Keeping the pre-processing passes high will increase render time but will also clear some artifacts and over-occlusions.

Error
: This is the tolerance factor for approximation error (i.e. the max allowed difference between approximated result and fully computed result). The lower, the slower the render, but the more accurate the results… Ranges between **0.0** and **10.0**, defaults to **0.250**.

Pixel Cache
: When enabled, it will keep values of computed pixels to interpolate it with its neighbors. This further speeds up the render, generally without visible loss in quality…

Correction
: A correction factor to reduce over-occlusion. Ranges between **0.0** (no correction) to **1.0**.

Exposure and Range

Mode: All modes

Panel: World (Shading context, World sub-context)

Description

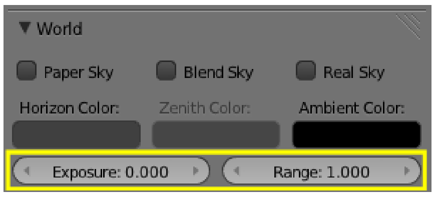
Exposure and Range are similar to the “Color Curves” tool in Gimp or Photoshop.

These controls affect the rendered image, and the results are baked into the render. For information on achieving similar affects with render controls, see Color Management and Exposure.

Previously Blender clipped color directly with “**1.0**” (or 255) when it exceeded the possible RGB space. This caused ugly banding and overblown highlights when light overflowed (*An overexposed teapot*).

Using an exponential correction formula, this now can be nicely corrected.

Options



Exposure: 0.000
Range: 1.000

Exposure and Range sliders.

Exposure
: The exponential curvature, with **0.0** being linear, and **1.0** being curved.

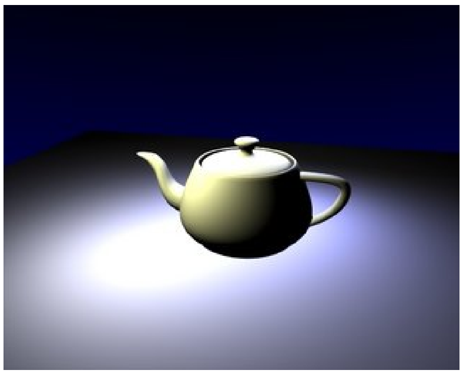
Range
: The range of input colors that are mapped to visible colors (**0.0** – **1.0**).

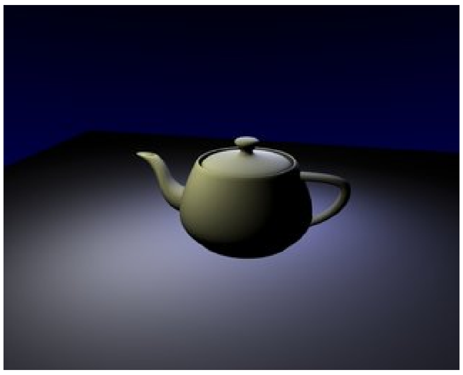
So without Exposure we will get a linear correction of all color values:

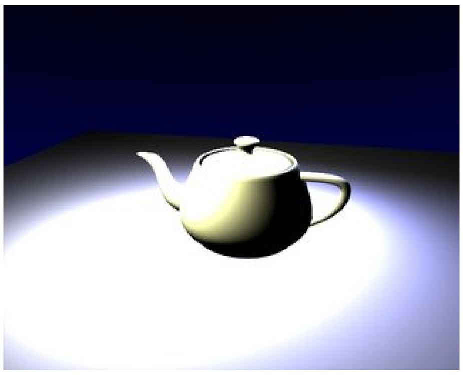
1. Range > **1.0**: the picture will become darker; with Range = **2.0**, a color value of **1.0** (the brightest by default) will be clipped to **0.5** (half bright) (*Range: **2.0***).
2. Range < **1.0**: the picture will become brighter; with Range = **0.5**, a color value of **0.5** (half bright by default) will be clipped to **1.0** (the brightest) (*Range: **0.5***).

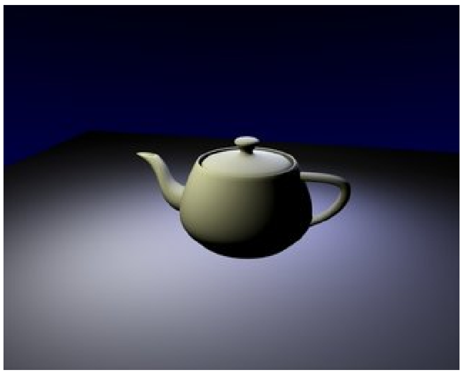
Examples

With a linear correction every color value will get changed, which is probably not what we want. Exposure brightens the darker pixels, so that the darker parts of the image won’t be changed at all (*Range: **2.0**, Exposure: **0.3***).


An overexposed teapot.


Range: **2.0**.


Range: **0.5**.

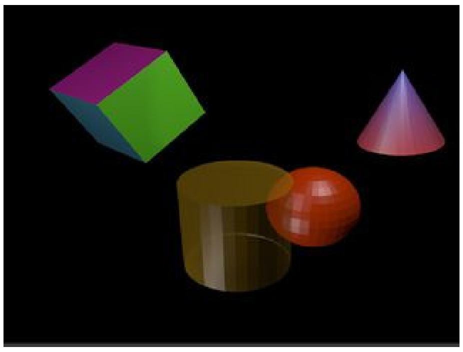

Range: **2.0**, Exposure: **0.3**.

Hints

Try to find the best Range value, so that overexposed parts are barely not too bright. Now turn up the Exposure value until the overall brightness of the image is satisfying. This is especially useful with area lamps.

Introduction to Materials

A material defines the artistic qualities of the substance that an object is made of. In its simplest form, you can use materials to show the substance an object is made of, or to "paint" the object with different colors. Usually, the substance is represented by its surface qualities (color, shininess, reflectance, etc.) but it can also exhibit more complicated effects such as transparency, diffraction and sub-surface scattering. Typical materials might be brass, skin, glass, or linen.


Various basic materials (single, multiple material, transparency, vertex paint).

The basic (un-textured) Blender material is uniform across each face of an object (although the various pixels of each face of the object may appear differently because of lighting effects). However, different faces of the object may use different materials (see Multiple Materials).

In Blender, materials can (optionally) have associated textures. Textures describe the substance: e.g. polished brass, dirty glass or embroidered linen. The Textures chapter describes how to add textures to materials.

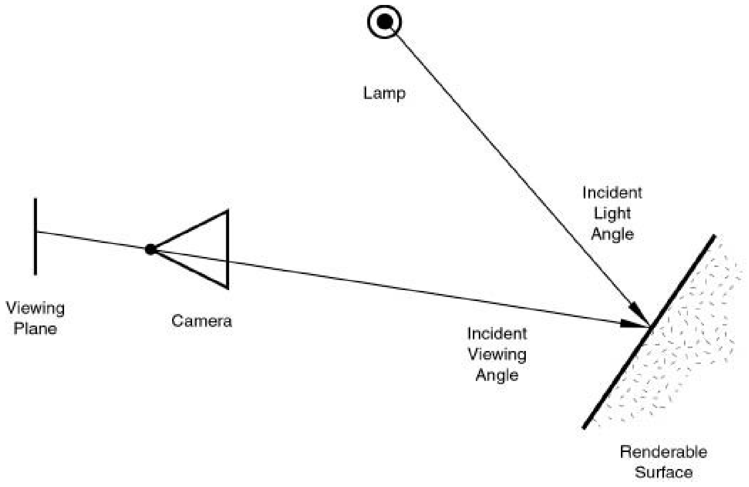
How Materials Work

Before you can understand how to design effectively with materials, you must understand how simulated light and surfaces interact in Blender's rendering engine and how material settings control those interactions. A deep understanding of the engine will help you to get the most from it.

The rendered image you create with Blender is a projection of the scene onto an imaginary surface called the *viewing plane*. The viewing plane is analogous to the film in a traditional camera, or the rods and cones in the human eye, except that it receives simulated light, not real light.

To render an image of a scene we must first determine what light from the scene is arriving at each point on the viewing plane. The best way to answer this question is to follow a straight line (the simulated light ray) backwards through that point on the viewing plane and the focal point (the location of the camera) until it hits a renderable surface in the scene, at which point we can determine what light would strike that point.

The surface properties and incident light angle tell us how much of that light would be reflected back along the incident viewing angle (*Rendering engine basic principle.*).



Rendering engine basic principle.

Two basic types of phenomena take place at any point on a surface when a light ray strikes it: diffusion and specular reflection. Diffusion and specular reflection are distinguished from each other mainly by the relationship between the incident light angle and the reflected light angle.

The shading (or coloring) of the object during render will then take into account the base color (as modified by the diffusion and specular reflection phenomenon) and the light intensity.

Using the internal ray tracer, other (more advanced) phenomena could occur. In ray-traced reflections, the point of a surface struck by a light ray will return the color of its surrounding environment, according to the rate of reflection of the material (mixing the base color and the surrounding environment's) and the viewing angle.

On the other hand, in ray-traced refractions, the point of a surface struck by a light ray will return the color of its background environment, according to the rate of transparency (mixing the base color and the background environment's along with its optional filtering value) of the material and the optional index of refraction of the material, which will distort the viewing angle.

Of course, shading of the object hit by a light ray will be about mixing all these phenomena at the same time during the rendering. The appearance of the object, when rendered, depends on many inter-related settings:

- World (Ambient color, Radiosity, Ambient Occlusion)
- Lights
- Material settings (including ambient, emission, and every other setting on every panel in that context)
- Texture(s) and how they are mixed
- Material Nodes
- Camera
- Viewing angle
- Obstructions and transparent occlusions
- Shadows from other opaque/transparent objects
- Render settings
- Object dimensions (SS settings are relevant to dimensions)
- Object shape (refractions, fresnel effects)

Using Materials

Check your Render

When designing materials (and textures and lighting), frequently check the rendered appearance of your scene, using your chosen render engine/shader settings. The appearance might be quite different from that shown in the texture display in the 3D panel.

As stated above, the material settings usually determine the surface properties of the object. There are several ways in which materials can be set up in Blender. Generally speaking, these are not compatible - you must choose which method you are going to use for each particular object in your scene.

First, you can set the Properties in the various Material panels.

Second, you can use Nodes; a graphical nodes editor is available.

Last, you can directly set the color of object surfaces using various special effects. Strictly speaking, these are not materials at all, but they are included here because they affect the appearance of your objects. These include Vertex Painting, Wire Rendering, Volume Rendering, and Halo Rendering.

The exact effect of Material settings can be affected by a number of system settings. First and foremost is the Render Engine used - Cycles and the Blender Render Engine (aka Blender Internal or BI) require quite different illumination levels to achieve similar results, and even then the appearance of objects can be quite different. Also, the material properties settings can be affected by the texture method used (Single Texture, Multitexture or GLSL). So it is recommended to always select the appropriate system settings before starting the design of materials.

Assigning a Material

Materials available in the currently-open Blender file can be investigated by clicking on the Materials button in the Properties Window Header. In this section we look at how to assign or remove a material to/from the Active Object in Blender, either by:

- creating a new material,
- re-using an existing material, or
- deleting a material.

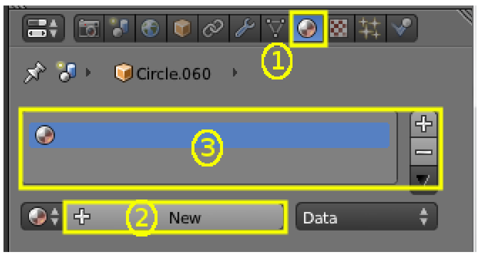
We also give hints about practical material usage.

Creating a new Material

Every time a new Object is created it has no material linked to it. You can create a new material for the object by

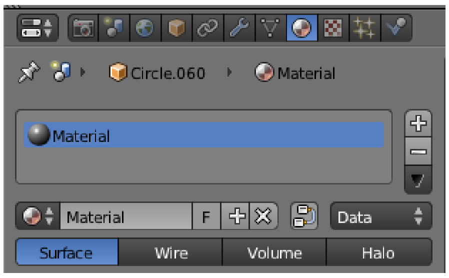
- Selecting the object
- In the Properties window, click on the object button
- Click on the Materials button in the Properties Panel Header (1)

The Shading context window then appears. This contains the following elements:



- Context - The currently-selected scene and object
- Object Material Slots (3) - this window shows the "slots" for the material (or materials) that this object data contains.
- Active Material (2). Initially empty, asking for "New".

To add a new material, click "+" in the Active Material box. This action has a series of effects:



- opens the new material in the Active Material box,
- brings up additional buttons in the immediate panel,
- adds the new material to the Available Materials list,
- adds the new material to the Object Material Slots list for the active object (or its object data - see below)
- brings up a preview of the new material,
- provides you with a range of panels allowing you to select the properties of the new material.

New Material Panel Buttons

Details of the additional buttons which appear in the Material panel for a new Active Material are as follows:

Active Material

- Available Materials
    See Reusing Existing Materials below.

Name
    Like other datablocks, Blender will automatically set the name of the new material to Material, Material.001 and so on. You can change this by over-typing with your own choice of name.

Number of Users
    Specifies the number of meshes which use this material.

F - Fake User;
    If lit, this material will always be saved within the Blender file, even if it has no meshes which use it (see Deleting a Material).

X
    Delete this material (see Deleting a Material).

Naming materials

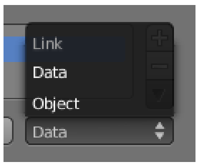
It's a very good idea to give your materials clear names so you can keep track of them, especially when they're linked to multiple objects. Try to make your names descriptive of the material, not its function (e.g. "Yellow Painted" rather than "Kitchen Table Color")

Nodes

If dark, use the Shader Nodes to generate the material.

Data

Specifies whether the material is to be linked to the Object or to the Object Data.



Link material to object or to object's data

The Link pop-up menu has two choices, Data and Object. These two menu choices determine whether the material is linked to the object or to the data, (in this case) the mesh (or curve, nurbs, etc.). The Data menu item determines that this material will be linked to the mesh's datablock which is linked to the object's datablock. The Object menu item determines that the material will be linked to the object's data block directly.
This has consequences of course. For example, different objects may share the same mesh datablock. Since this datablock defines the shape of the object, any change in edit mode will be reflected on all of those objects. Moreover, anything linked to that mesh datablock will be shared by every object that shares that mesh. So, if the material is linked to the mesh, every object will share it.
On the other hand, if the material is linked directly to the object datablock, the objects can have different materials and still share the same mesh. Short explanation: If connected to the object, you can have several instances of the same obData using different materials. If linked to mesh data, you can't. See Data System for more information.

Object Render Format menu.

This menu has four options which define how the object is to be rendered:

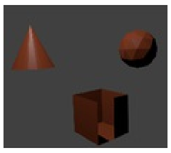
Surface
Material applied to object planes.

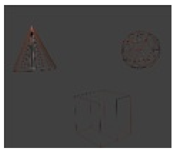
Wire
Material applied to wires following the object edges

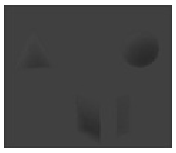
Volume
Material applied to the object volume.

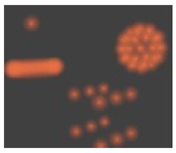
Halos
Material applied to halos around each object vertex.

Object Renders


Surface


Wire

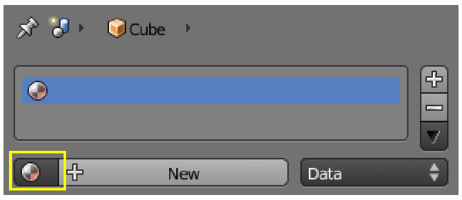

Volume


Halo

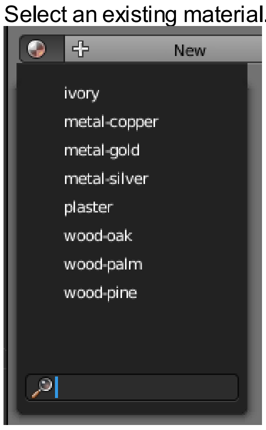
Reusing Existing Materials

Blender is built to allow you to reuse *anything*, including material settings, between many objects. Instead of creating duplicate materials, you can simply re-use an existing material. There are several ways to do this using the Available Materials menu:

Single Object - With the object selected, click the sphere located to the left of the Material name. A drop-down list appears showing all the materials available in the current Blender file. To use one, just click on it.



Select an existing material.



List of available materials

Searching for Materials

The search field at the bottom of the material list allows you to search the names in the list. For example, by entering "wood" all existent materials are filtered so that only materials containing "wood" are displayed in the list.

Multiple Objects - In the 3D View, with CtrlL you can quickly link all selected objects to the material (and other aspects) of the active object. Very useful if you need to set a large number of objects to the same material; just select all of them, then the object that has the desired material, and CtrlL link them to that "parent". (See Tip on Linking Data in Creating about data linking.)

Deleting a Material

To delete a material, select the material and click X in the Available Materials List entry.

Although the material will seem to disappear immediately, the Delete action can depend on how the material is used elsewhere.

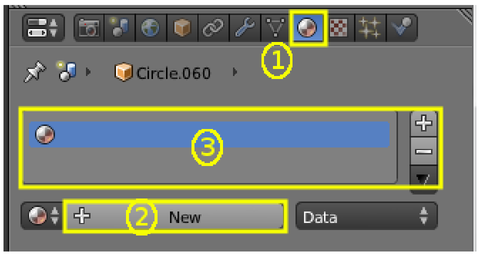
If the material is linked to the Object and there are other objects which use this material, then the material will be removed from that object (but remain on all its other objects).

If the "Fake User" button (F) has been lit in the Available Materials list, then the material will be retained when the file is saved, even if it has no users.

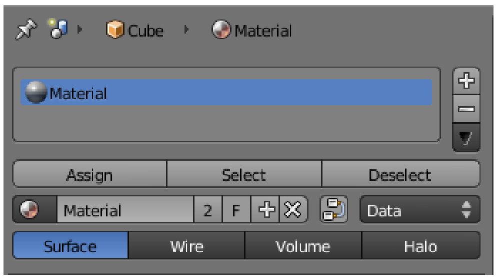
Only if it has 0 "real" users, and no "Fake" user, will the material be permanently deleted. Note that it will still remain in the Materials list until the Blender file is saved, but will have disappeared when the file is reloaded.

Multiple Materials

Normally, different colors or patterns on an object are achieved by adding textures to your materials. However, in some applications you can obtain multiple colors on an object by assigning different materials to the individual faces of the object.



To apply several materials to different faces of the same object, you use the Material Slots options (3) in the Materials header panel.



Material menu in edit mode

The workflow for applying a second material to some faces of an object covered by a base material is as follows:

- In Object mode, apply the base material to the whole object (as shown in Assigning a material)
- Create/select the second material (the whole object will change to this new material).
- In the Active Material box (2), re-select the base material.
- Go to Edit Mode - Face Select (a new box appears above the Active Material box with Assign/Select/Deselect).
- Select the face/faces to be colored with the second material.
- In the Object Material Slots box (3), click the + to create a new slot, and while this is still active, click on the second material in the Available Materials list.
- Click the Assign button, and the second material will appear on the selected object faces.

- You can also make this new material a copy of an existing material by adding the data block:

Select object, get the material, (R Click) – Copy data to clipboard. When you have renamed the material, click “Data – Data” to link to the existing material. Proceed to assign faces as required. NB: If you change the material on the original object, the new object color changes too.

Introduction to Properties

Material Properties

Materials can have a wide array of properties. It is the combination of all of these things that define the way a material looks, and how objects using that material will appear when rendered. These properties are set using the various Properties panels.

Remember that the appearance of your materials are affected by the way that they are rendered (surface, wire, volume or halo), and by the rendering engine (Blender, Cycles, or Game) used. Most properties for images rendered using Cycles can only be controlled using the Node system.

The table below sets out the various Properties panels available, which Render Engine they are available for, and brief details of their scope. Details of their controls and settings are given on the relevant pages.

Material Property Panels

Panel	**R**	**G**	**Description**
Preview	X	X	A preview of the current material mapped on to one of several basic objects.
Diffuse Shaders	X	X	The basic color of the material, together with different models for dispersion.
Specular Shaders	X	X	The reflected highlights: color, strength and different models for dispersion.
Color Ramps	X	X	How to vary the base color over a surface in both Diffuse ans Specular shaders.
Shading	X	X	Properties of various characteristics of the shading model for the material.
Transparency	X	(X)	Can other objects be seen through the object, and if so, how?
Mirror	X		Reflective properties of the material.
SubSurface Scattering	X		Simulates semi-translucent objects in which light enters, bounces around, then exits in a different place.
Strand	X		For use when surfaces are covered with hair, fur, etc.
Options	X	X	Various options for shading and coloring the object.
Shadow	X	X	Controls how objects using this material cast and receive shadows.
Game Settings		X	Controls settings for real-time rendering of Game Engine objects.
Physics		X	??

Render Engine Key: R = Blender Render; G = Game Engine

Material Preview

Mode: All Modes

Panel: Shading/Material Context → Preview

The Preview panel gives a quick visualization of the active material and its properties, including its Shaders, Ramps, Mirror Transp properties and Textures. It provides several shapes that are very useful for designing new shaders: for some shaders (like those based on Ramp colors, or a Diffuse shader like Minnaert), one needs fairly complex or specific previewing shapes to decide if the shader being designed achieves its goal.

Options

Flat XY plane
: Useful for previewing textures and materials of flat objects, like walls, paper and such.

Sphere
: Useful for previewing textures and materials of sphere-like objects, but also to design metals and other reflective/transparent materials, thanks to the checkered background.

Cube
: Useful for previewing textures and materials of cube-like objects, but also to design procedural textures. Features a checkered background.

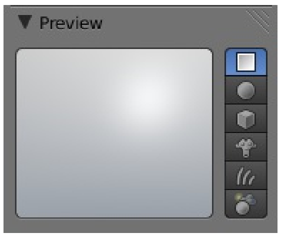
Monkey
: Useful for previewing textures and materials of organic or complex non-primitive shapes. Features a checkered background.

Hair strands
: Useful for previewing textures and materials of strand-like objects, like grass, fur, feathers and hair. Features a checkered background.

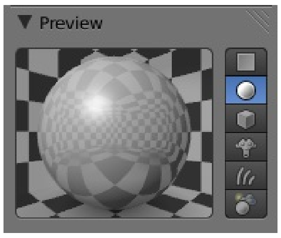
Large Sphere with Sky
: Useful for previewing textures and materials of sphere-like objects, but also to design metals and other reflective materials, thanks to the gradient Sky background.

Preview uses OSA (oversampling). Whatever the preview option, it will make use of OSA (oversampling) in order to provide better quality. Disable this option if your computer is already slow or old.

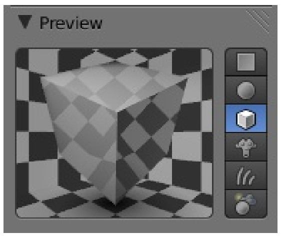
Examples



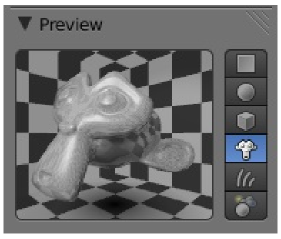
Plane preview.



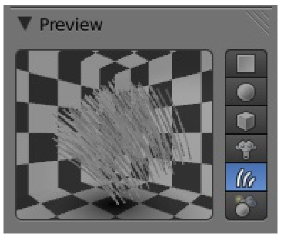
Sphere preview.



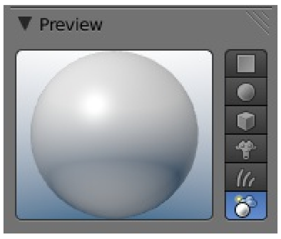
Cube preview.



Monkey preview.



Hair Strands preview.



Sky Sphere preview.

Diffuse Shaders

Mode: All Modes

Panel: Shading/Material Context → Diffuse

A diffuse shader determines, simply speaking, the general color of a material when light shines on it. Most shaders that are designed to mimic reality give a smooth falloff from bright to dark from the point of the strongest illumination to the shadowed areas, but Blender also has other shaders for various special effects.

Common Options

All diffuse shaders have the following options:

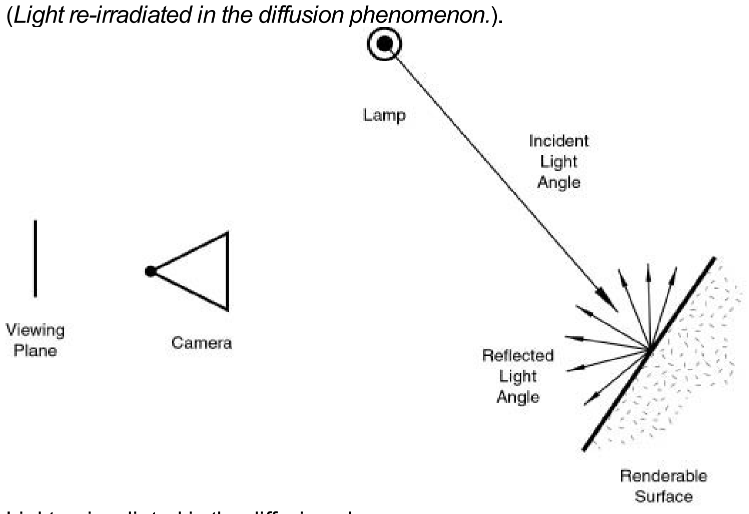
Color
: Select the base *diffuse color* of the material.

Intensity
: The shader's brightness, or more accurately, the amount of incident light energy that is actually diffusely reflected towards the camera.

Ramp
: Allows you to set a range of colors for the Material, and define how the range will vary over a surface. See Color Ramps for details.

Technical Details

Light striking a surface and then re-irradiated via a Diffusion phenomenon will be scattered, i.e., re-irradiated in all directions isotropically. This means that the camera will see the same amount of light from that surface point no matter what the *incident viewing angle* is. It is this quality that makes diffuse light *viewpoint independent*. Of course, the amount of light that strikes the surface depends on the incident light angle. If most of the light striking a surface is reflected diffusely, the surface will have a matte appearance (*Light re-irradiated in the diffusion phenomenon.*).



Light re-irradiated in the diffusion phenomenon.

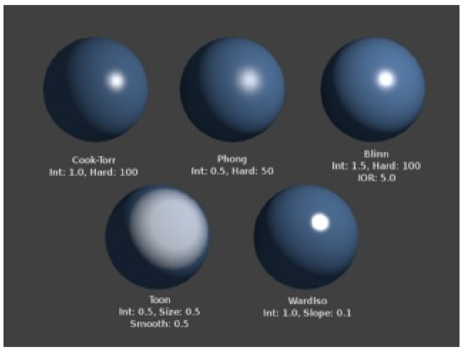
Shader Names

Some shaders' names may sound odd - they are traditionally named after the people who first introduced the models on which they are based.

Lambert

Mode: All Modes

Panel: Shading/Material Context → Shaders

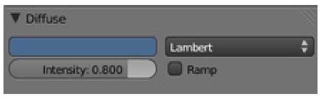


Lambert Shader

This is Blender's default diffuse shader, and is a good general all-around workhorse for materials showing low levels of specular reflection.

Johann Heinrich Lambert (1728-1777)
: was a Swiss mathematician, physicist and astronomer who published works on the reflection of light, most notably the Beer-Lambert Law which formulates the law of light absorption.

This shader has only the default option, determining how much of available light is reflected. Default is 0.8, to allow other objects to be brighter.

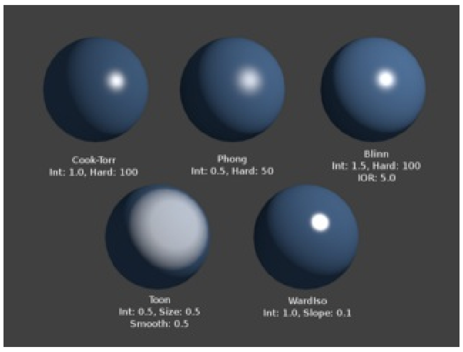

Diffuse
Lambert
Intensity: 0.800
Ramp

The Lambert diffuse shader settings.

Oren-Nayar

Mode: All Modes

Panel: Shading/Material Context → Shaders



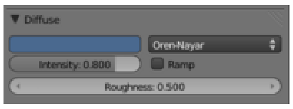
Oren-Nayar Shader

Oren-Nayar takes a somewhat more 'physical' approach to the diffusion phenomena as it takes into account the amount of microscopic roughness of the surface.

Michael Oren and Shree K. Nayar
: Their reflectance model, developed in the early 1990s, is a generalization of Lambert's law now widely used in computer graphics.

Options

Roughness
: The roughness of the surface, and hence, the amount of diffuse scattering.

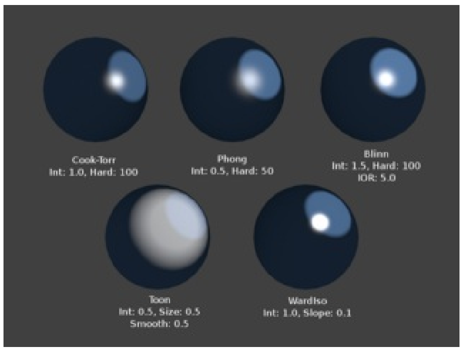

Diffuse
Oren-Nayar
Intensity: 0.800
Ramp
Roughness: 0.500

The Oren-Nayar diffuse shader settings.

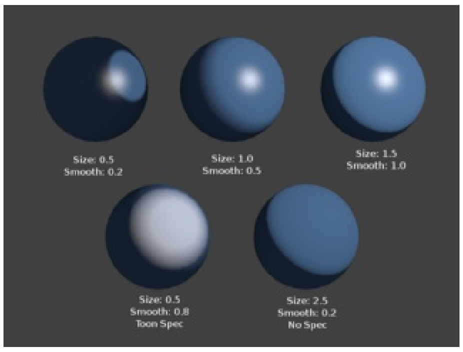
Toon

Mode: All Modes

Panel: Shading/Material Context → Shaders



Toon Shader, Different Spec



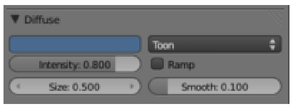
Toon Shader Variations

The Toon shader is a very 'un-physical' shader in that it is not meant to fake reality but to produce cartoon cel styled rendering, with clear boundaries between light and shadow and uniformly lit/shadowed regions.

Options

Size
: The size of the lit area

Smooth
: The softness of the boundary between lit and shadowed areas

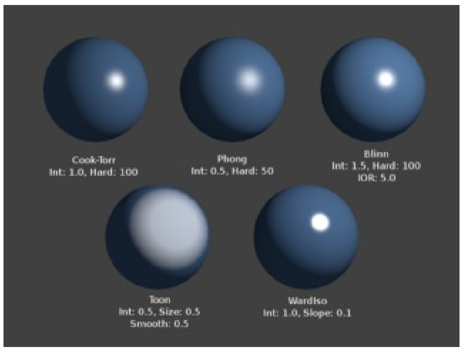

▼ Diffuse
Toon
Intensity: 0.800
Ramp
Size: 0.500
Smooth: 0.100

The Toon diffuse shader settings.

Minnaert

Mode: All Modes

Panel: Shading/Material Context → Shaders



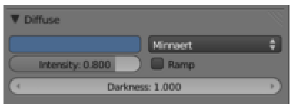
Minnaert Shader

Minnaert works by darkening parts of the standard Lambertian shader, so if *Dark* is 1 you get exactly the Lambertian result. Higher darkness values will darken the center of an object (where it points towards the viewer). Lower darkness values will lighten the edges of the object, making it look somewhat velvet.

Marcel Minnaert (1893-1970)
: was a Belgian astronomer interested in the effects of the atmosphere on light and images who in 1954 published a book entitled *The Nature of Light and Color in the Open Air*.

Options

Dark
: The darkness of the 'lit' areas (higher) or the darkness of the edges pointing away from the light source (lower).

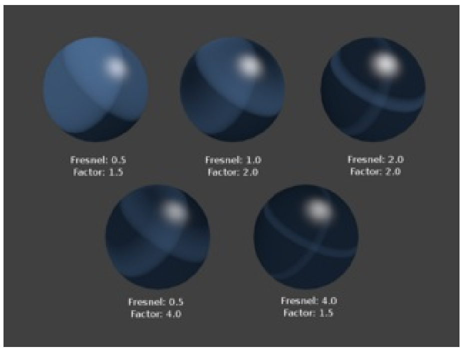

▼ Diffuse
Minnaert
Intensity: 0.800
Ramp
Darkness: 1.000

The Minnaert diffuse shader settings.

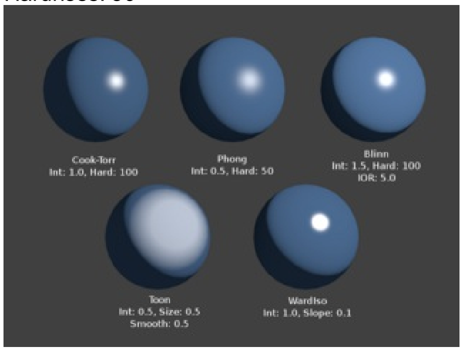
Fresnel

Mode: All Modes

Panel: Shading/Material Context → Shaders



Various settings for the Fresnel shader, Cook-Torr Specular shader kept at Intensity 0.5, Hardness: 50



Fresnel Shader, Different Spec

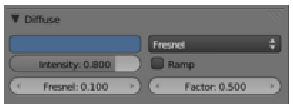
With a Fresnel shader the amount of diffuse reflected light depends on the incidence angle, i.e. from the direction of the light source. Areas pointing directly towards the light source appear darker; areas perpendicular to the incoming light become brighter.

Augustin-Jean Fresnel (1788-1827)
: was a French physicist who contributed significantly to the establishment of the theory of wave optics.

Options

Fresnel
: Power of the Fresnel effect, 5.0 is max.

Factor
: Blending factor of the Fresnel factor to blend in, 5.0 is max.


▼ Diffuse
Fresnel
Intensity: 0.800
Ramp
Fresnel: 0.100
Factor: 0.500

The Fresnel diffuse shader settings.

Specular Shaders

Mode: All Modes

Panel: Shading/Material Context → Specular

Specular shaders create the bright highlights that one would see on a glossy surface, mimicking the reflection of light sources. Unlike diffuse shading, specular reflection is *viewpoint dependent*. According to Snell's Law, light striking a specular surface will be reflected at an angle which mirrors the incident light angle (with regard to the surface's normal), which makes the viewing angle very important.

Not a Mirror!

It is important to stress that the *specular reflection* phenomenon discussed here is not the reflection we would see in a mirror, but rather the light highlights we would see on a glossy surface. To obtain true mirror-like reflections you would need to use the internal raytracer. Please refer to section RENDERING of this manual.

Common Options

Each specular shader share the following common options:

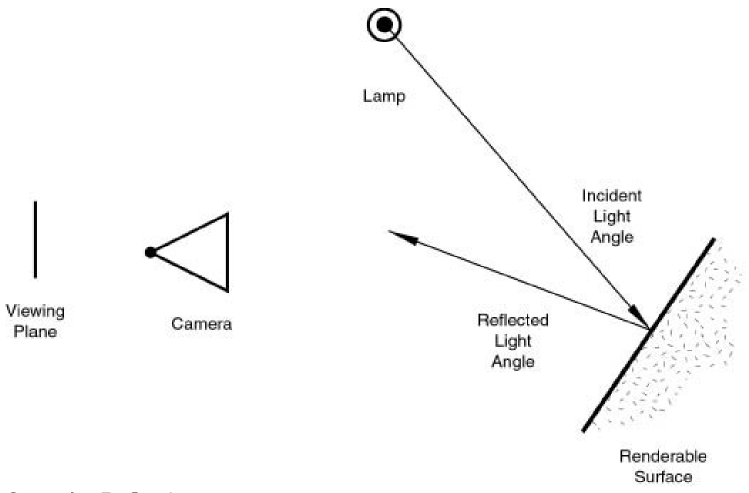
Specular color
: The color of the specular highlight

Intensity
: The intensity, or brightness of the specular highlight. This has a range of [0-1].

Ramp
: Allows you to set a range of specular colors for Material, and define how the range will vary over a surface. See Ramps for details.

As a result, a material has at least two different colors, a diffuse, and a specular one. The specular color is normally set to pure white (the same "pure white" as the reflected light source), but it can be set to different values for various effects (e.g. metals tend to have colored highlights).

Technical Details



Specular Reflection.

In reality, the quality of Diffuse and Specular reflection are generated during the same process of light scattering, but are not the same. Diffusion is actually subsurface scattering at a very small scale.

Imagine that a surface is made up of extremely microscopic semi-transparent, reflective facets. The sharpness of Specular reflection is determined by the distribution of the angle of these microfacets on the surface of an object. The more deep and jagged these facets are, the more the light spreads when it hits the surface. When these facets are flatter against the "macrosurface", the surface will have a tighter reflection, closer to a mirror. This is a condensed explanation of the generally accepted microfacet theory of reflectance, which is the basis of all modern BRDFs (Bi-directional Reflectance Distribution Functions), or shading models.

Because these microfacets are transparent, some light that hits them travels into the surface and diffuses. The light that makes it back out is roughly Lambertian most of the time, meaning that it spreads evenly in all directions. It is also attenuated by the pigmentation in the surface, hence creating what we perceive as diffuse, and the color of an object.

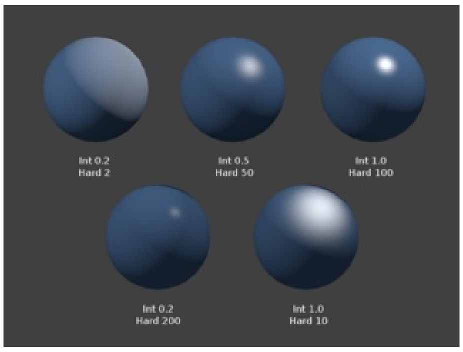
Note that at glancing angles, the reflectivity of a surface will always go to 1.

If it is difficult for you to understand this relationship, try to imagine a ball (say, of centimeter scale): if you throw it against a wall of raw stones (with a scale of roughness of a decimeter), it will bounce in a different direction each time, and you will likely quickly lose it! On the other hand, if you throw it against a smooth concrete wall (with a roughness of, say, a millimeter scale), you can quite easily anticipate its bounce, which follow (more or less!) the same law as the light reflection…

CookTorr

Mode: All Modes

Panel: Shading/Material Context → Shaders



CookTorr Shader (Lambert 0.8)

CookTorr (Cook-Torrance) is a basic specular shader that is most useful for creating shiny plastic surfaces. It is a slightly optimized version of Phong.

Robert L. Cook (LucasFilm) and Kenneth E. Torrance (Cornell University)
: In their 1982 paper *A Reflectance Model for Computer Graphics* (PDF), they described "a new reflectance model for rendering computer synthesized images" and applied it to the simulation of metal and plastic.

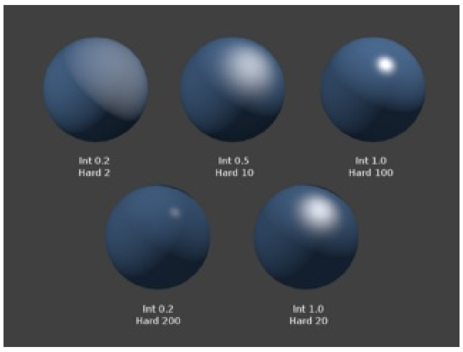
Options

Hardness
: Size of the specular highlight

Phong

Mode: All Modes

Panel: Shading/Material Context → Shaders



Phong Shader (Lambert 0.8)

Phong is a basic shader that's very similar to CookTorr, but is better for skin and organic surfaces.

Bui Tuong Phong (1942-1975)
: was a Vietnamese-born computer graphics pioneer that developed the first algorithm for simulating specular phenomenon. His model included components not only for specular lighting, but also diffuse and ambient lighting.

Options

Hardness
: Size of the specular highlight.

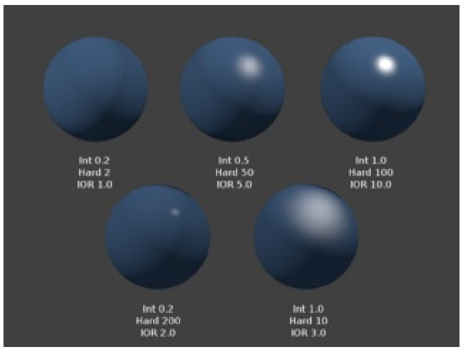
Planet Atmosphere

Because of its fuzziness, this shader is good for atmosphere around a planet. Add a sphere around the planet, slightly larger than the planet. For its material, use a phong specular shader. Set it to a very low alpha (.05), zero diffuse, low hardness (5) but high specularity (1).

Blinn

Mode: All Modes

Panel: Shading/Material Context → Shaders



Blinn Shader (Oren-Nayar Int 0.8, Rough 0.5)

Blinn is a more 'physical' specular shader, often used with the Oren-Nayar diffuse shader. It can be more controllable because it adds a fourth option, an *index of refraction* (IOR), to the aforementioned three.

James F. Blinn
: worked at NASA's Jet Propulsion Laboratory and became widely known for his work on Carl Sagan's TV documentary *Cosmos*. The model he described in his 1977 paper *Models of Light Reflection for Computer Synthesized Pictures* (PDF) included changes in specular intensity with light direction and more accurately positioned highlights on a surface.

Options

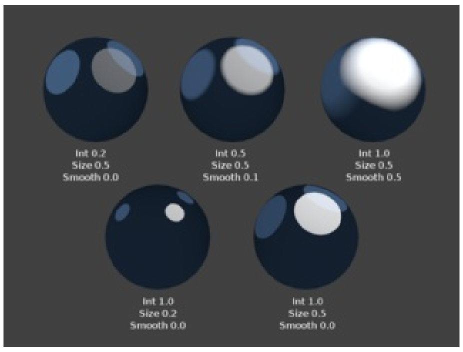
Hardness
: Size of the specular highlight. The Blinn shader is capable of much tighter specular highlights than Phong or CookTorr.

IOR
: 'Index of Refraction'. This parameter is not actually used to compute refraction of light rays through the material (a ray tracer is needed for that), but to correctly compute specular reflection intensity and extension via Snell's Law.

Toon

Mode: All Modes

Panel: Shading/Material Context → Shaders



Toon Specular Shader (Toon Diffuse, Int 0.8, Size & Smooth match)

The Toon specular shader matches the Toon diffuse shader. It is designed to produce the sharp, uniform highlights of cartoon cels.

Options

Size
: Size of the specular highlight.

Smooth
: Softness of the highlight's edge.

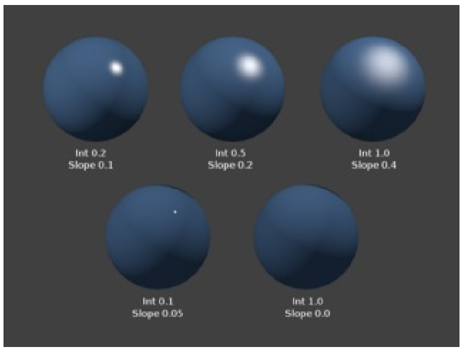
Alternative Method

The Toon shader effect can also be accomplished in a more controllable way using ColorRamps.

WardIso

Mode: All Modes

Panel: Shading/Material Context → Shaders



WardIso Shader

WardIso is a flexible specular shader that can be useful for metal or plastic.

Gregory J. Ward
: developed a relatively simple model that obeyed the most basic laws of physics. In his 1992 paper, *Measuring and modeling anisotropic reflection,* Ward introduced a Bidirectional Reflectance Distribution Function (BRDF) since then widely used in computer graphics because the few parameters it uses are simple to control. His model could represent both isotropic surfaces (independent of light direction) and anisotropic surfaces (direction dependent). In Blender, the Ward specular shader is still called **Ward Isotropic** but is actually anisotropic. (PDF)

Options

Slope
: Standard deviation for of surface slope. Previously known as the root-mean-square or rms value, this parameter in effect controls the size of the specular highlight, though using a different method to that of the other specular shaders. It is capable of extremely sharp highlights.

Color Ramps

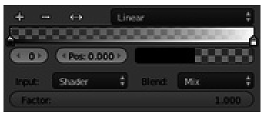
Mode: All Modes

Panel: Context Shading → sub-context Material → Ramps

In many real life situations — like skin or metals — the color of diffuse and specular reflections can differ slightly, based on the amount of energy a surface receives or on the light angle of incidence. The Ramp Shader options in Blender allow you to set a range of colors for a Material, and define how the range will vary over a surface, and how it blends with the 'actual color' (typically from a material or as output of a texture).
Ramps allow you to precisely control the color gradient across a material, rather than just a simple blend from a brightened color to a darkened color, from the most strongly lit area to the darkest lit area. As well as several options for controlling the gradient from lit to shadowed, ramps also provide 'normal' input, to define a gradient from surfaces facing the camera to surfaces facing away from the camera. This is often used for materials like some types of metallic car paint that change color based on viewing angle.

Since texture calculations in Blender happen before shading, the Ramp Shader can completely replace texture or material color. But by use of the mixing options and Alpha values it is possible to create an additional layer of shading in Blender materials.

Options



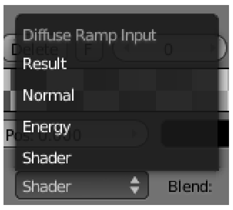
Ramps Panel

In Blender 2.5, the separate Ramp panels for the Diffuse shader and the Specular shader respectively can be toggled on and off using the ☑ Ramp button.

By default the Ramp panel opens with two colors; the first stop (0) is black and transparent (Alpha=0) and the second stop (1) is white and opaque (Alpha=1).

The position of the color stop markers can be altered by either (1) dragging the stop marker in the colorband or (2) by changing the Pos value in the Pos: 0.000 box.

Color and alpha values for each marker can be set by clicking the box.



Input popup menu

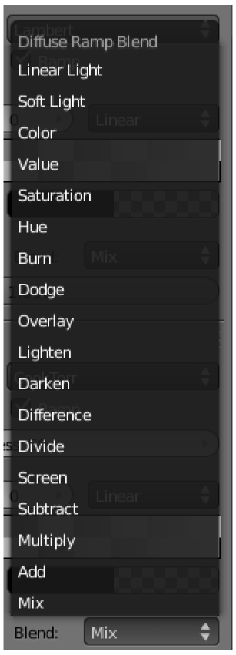
Input
: The input menu contains the following options for defining the gradient:

    Shader
    : The value as delivered by the material's shader (Lambert, CookTorr) defines the color. Here the amount of light doesn't matter for color, only the direction of the light.

    Energy
    : As Shader, now also lamp energy, color, and distance are taken into account. This makes the material change color when more light shines on it.

    Normal
    : The surface normal, relative to the camera, is used for the Ramp Shader. This is possible with a texture as well, but added for convenience.

    Result
    : While all three previous options work per lamp, this option only works after shading calculations. This allows full control over the entire shading, including 'Toon' style results. Using alpha values here is most useful for tweaking a finishing touch to a material.



Blend popup menu

Blend
: A list of the various blending modes available for blending the ramp shader with the color from Input.

Factor
: This slider denotes the overall factor of the ramp shader with the color from Input.

Colorbands

Mode: All Modes

Panel: Context Shading → sub-context Material → Ramps

A colorband can contain a gradient through a sequence of many colors (with alpha), each color acting across a certain position in the spectrum. Colorbands are used in both materials and textures, as well in other places where a range of colors can be computed and displayed.

Options

Add
: Add a new mark to the center of the colorband with the default color (neutral gray). New marks can also be added by Ctrl LMB 🖱 clicking in the colorband itself, which will add the mark at the position of the click with the same color that already exists underneath the mouse pointer.

Delete
: Remove the currently selected mark from the colorband.

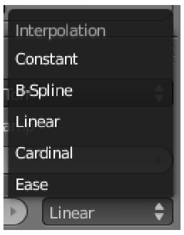
F
: Flip the colorband.

0
: The number of the active mark. The values for this mark are those being displayed, and in the colorband, the active mark is displayed as a dashed line. Another marker can be selected (1) using the arrows in the ◂ 0 ▸ slider, (2) by clicking on the number being displayed and entering a number of a color mark, or (3) by LMB 🖱 clicking a marker in the colorband.

Pos
: The position of the active color mark in the colorband (range 0.0–1.0). The position of the color marks can also be changed by LMB 🖱 dragging them in the colorband.

Reordering colors
If the position of the color marks are reordered, they will be automatically renumbered so that they always start with **0** from the left and increment to the right.

The Colorswatch right of the Position slider displays the color of the active mark. LMB 🖱 click it to display a color picker in which values for color (RGB) and transparency (Alpha) can be set.



Interpolation
popup menu

Interpolation

Various modes of interpolation between marker's values can be chosen in the Interpolation menu:

Ease
Ease by quadratic equation.

Cardinal
Cardinal.

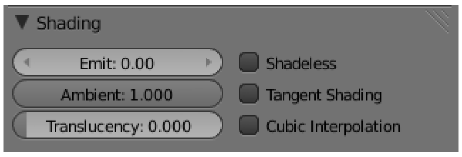
Linear
Linear (default). A smooth, consistent transition between colors.

B-Spline
B-Spline.

Constant
Constant.

Shading

In the separate Shading tab six more options are available:



Emit
Amount of light to emit

Ambient
Amount of global ambient color the material receives. Each material has an Ambient slider that lets you choose how much ambient light that object receives. Set to 1.0 by default.

You should set this slider depending on the amount of ambient light you think the object will receive. Something deep in the cave will not get any ambient light, whereas something close to the entrance will get more. Note that you can animate this effect, to change it as the object comes out of the shadows and into the light.

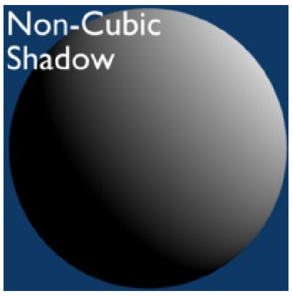
Settings for Ambient Occlusion and Environment Lighting can be found in the World menu, with parameters affecting both these lighting components found in the World Gather menu.

Translucency
Amount of diffuse shading on the back side

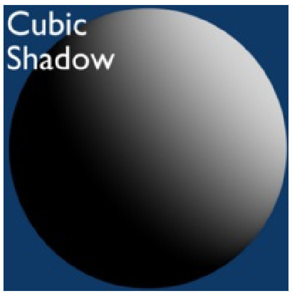
Shadeless
Make this material insensitive to light or shadow; gives a solid, uniform color to the whole object.

Tangent Shading
Use the material's tangent vector instead of the normal for shading — for anisotropic shading effects (e.g. soft hair and brushed metal). This shading was introduced in 2.42, see also settings for strand rendering in the menu further down and in the Particle System menu.

Cubic Interpolation
Use cubic interpolation for diffuse values. Enhances the contrast between light areas and shadowed areas



Without Cubic enabled.



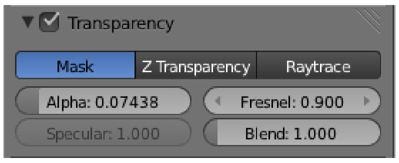
With Cubic enabled.

Transparency

Mode: All Modes

Panel: Shading/Material Context → Transparency

Materials in Blender can be set to be transparent, so that light can pass through any objects using the material. Transparency is controlled using an "alpha" channel, where each pixel has an additional value, range 0-1, in addition to its RGB color values. If alpha=0, then the pixel is transparent, and the RGB values for the surface contribute nothing to the pixel's appearance; for alpha=1, the surface is fully opaque, and the color of the surface determines the final color of the pixel.



In Blender, there are three ways in which the transparency of a material can be set: Mask, Z-Buffer and Ray-trace. Each of these is explained in more detail below. The Material Preview option with a sphere object gives a good demonstration of the capabilities of these three options.

Common Options

The following property controls are available for all transparency options:

Alpha
: Sets the transparency of the material by setting all pixels in the alpha channel to the given value.

Fresnel
: Sets the power of the Fresnel effect. The Fresnel effect controls how transparent the material is, depending on the angle between the surface normal and the viewing direction. Typically, the larger the angle, the more opaque a material becomes (this generally occurs on the outline of the object).

Specular -
: Controls the alpha/falloff for the specular color.

Blend
: Controls the blending between transparent and non-transparent areas. Only used if Fresnel is greater than 0.

Mask

This option simply masks the Background. It uses the alpha channel to mix the color of each pixel on the active object plane with the color of the corresponding background pixel, according to the alpha channel of the pixel. Thus for alpha = 1, the object color is seen - the object is completely opaque; but if alpha = 0, only the background is seen - the object is transparent (but note that any other object behind the active object disappears).

This is useful for making textures of solid or semi-transparent objects from photographic reference material - a mask is made with alpha opaque for pixels within the object, and transparent for pixels outside the object.

See Mask Transparency.

Z Buffer

This uses the alpha buffer for transparent faces. The alpha value of each pixel determines the mix of the basic color of the material, and the color of the pixel is determined from the objects/background behind it. Only basic settings are available with this option; it does not calculate refractions.

Raytraced Transparency

Uses ray tracing to calculate refractions. Ray tracing allows for complex refractions, falloff, and blurring, and is used for simulating the refraction of light rays through a transparent material, like a lens.

Note that the RayTrace option is only available in the Blender Render and Cycles render engines, but not in the Game Engine.

A ray is sent from the camera and travels through the scene until it encounters an object. If the first object hit by the ray is non-transparent, then the ray takes the color of the object.

If the object is transparent, then the ray continues its path through it to the next object, and so on, until a non-transparent object is finally encountered which gives the whole chain of rays its color. Eventually, the first transparent object inherits the colors of its background, proportional to its Alpha value (and the Alpha value of each transparent Material hit in between).

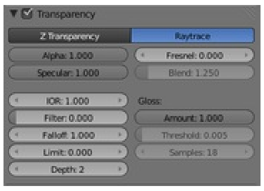
But while the ray travels through the transparent object, it can be deflected from its course according to the Index of Refraction (IOR) of the material. When you actually look through a plain sphere of glass, you will notice that the background is upside-down and distorted: this is all because of the Index of Refraction of glass.

Enable Raytracing

To get ray-traced transparency, you need to:

1. enable ray tracing in your Render settings. This is done in the Render context → Shading Panel. Ray tracing is enabled by default.
2. set your Alpha value to something other than 1.0.
3. in order for the background material to receive light passing through your transparent object, Receive Transparent must be turned on for that material in the Material → Shadow panel.

Options


Transparency
Z Transparency
Raytrace
Alpha: 1.000
Fresnel: 0.000
Specular: 1.000
Blend: 1.250
IOR: 1.000
Gloss:
Filter: 0.000
Amount: 1.000
Falloff: 1.000
Threshold: 0.005
Limit: 0.000
Samples: 18
Depth: 2

The Transparency Panel.

In addition to the common options given above, the following property controls are available:

IOR
: Index of Refraction. Sets how much a ray traveling through the material will be refracted, hence producing a distorted image of its background. See IOR values for Common Materials below.

Filter
: Amount of filtering for transparent ray trace. The higher this value, the more the base color of the material will show. The material will still be transparent but it will start to take on the color of the material. Disabled (0.0) by default.

Falloff
: How fast light is absorbed as it passes through the material. Gives 'depth' and 'thickness' to glass.

Limit
: Materials thicker than this are not transparent. This is used to control the threshold after which the filter color starts to come into play.

Depth
: Sets the maximum number of transparent surfaces a single ray can travel through. There is no typical value. Transparent objects outside the Depth range will be rendered pitch black if viewed through the transparent object that the Depth is set for. In other words, if you notice black areas on the surface of a transparent object, the solution is probably to increase its Depth value (this is a common issue with ray tracing transparent objects). You may also need to turn on transparent shadows on the background object.

Gloss
: Settings for the glossiness of the material.

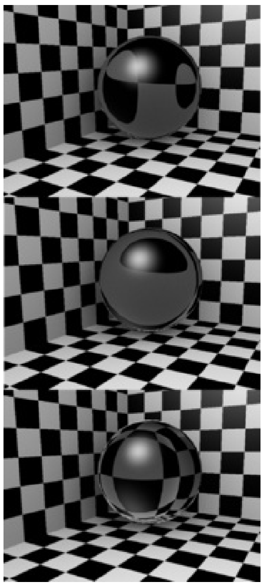
  Amount
  : The clarity of the refraction. Set this to something lower than zero to get a blurry refraction.

  Threshold
  : Threshold for adaptive sampling. If a sample contributes less than this amount (as a percentage), sampling is stopped.

  Samples
  : Number of cone samples averaged for blurry refraction.

Examples

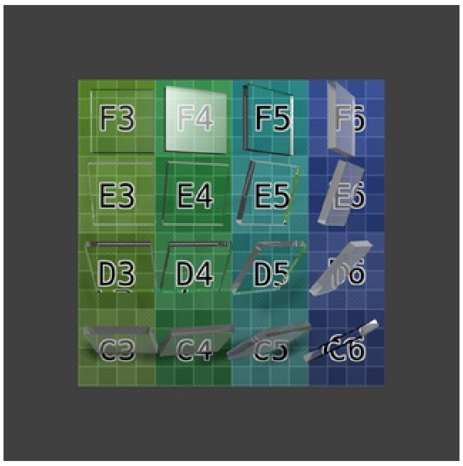
Index of Refraction

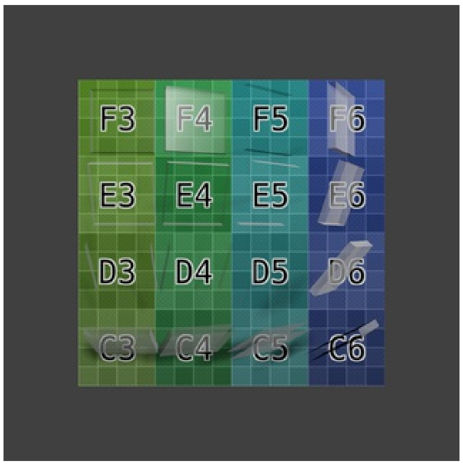

Influence of the IOR of an Object on the distortion of the background: spheres of Water, Glass and

Diamond (top to bottom).

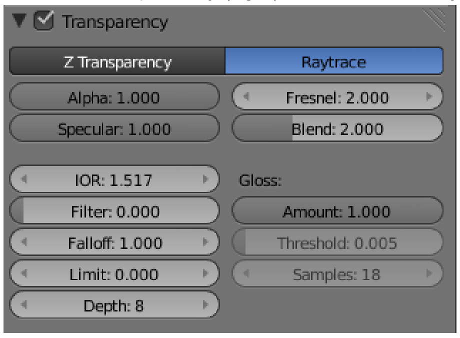
(*Influence of the IOR of an Object on the distortion of the background: spheres of Water, Glass and Diamond (top to bottom).*).
There are different values for typical materials: Air is **1.000** (no refraction), Alcohol is **1.329**, Glass is **1.517**, Plastic is **1.460**, Water is **1.333** and Diamond is **2.417**.

Fresnel





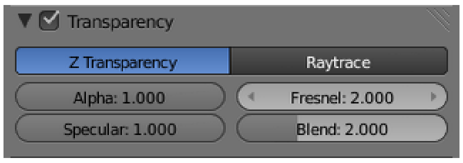
16 pieces of glass rotated in various directions demonstrate the angle-dependent Fresnel effect with ray-traced (left) and alpha buffered transparency (right). Note that the major difference is the lack of IOR effect in the latter case. (Download .blend.)



Transparency

Z Transparency | Raytrace

Alpha: 1.000
Fresnel: 2.000
Specular: 1.000
Blend: 2.000
IOR: 1.517
Gloss:
Filter: 0.000
Amount: 1.000
Falloff: 1.000
Threshold: 0.005
Limit: 0.000
Samples: 18
Depth: 8



Transparency

Z Transparency | Raytrace

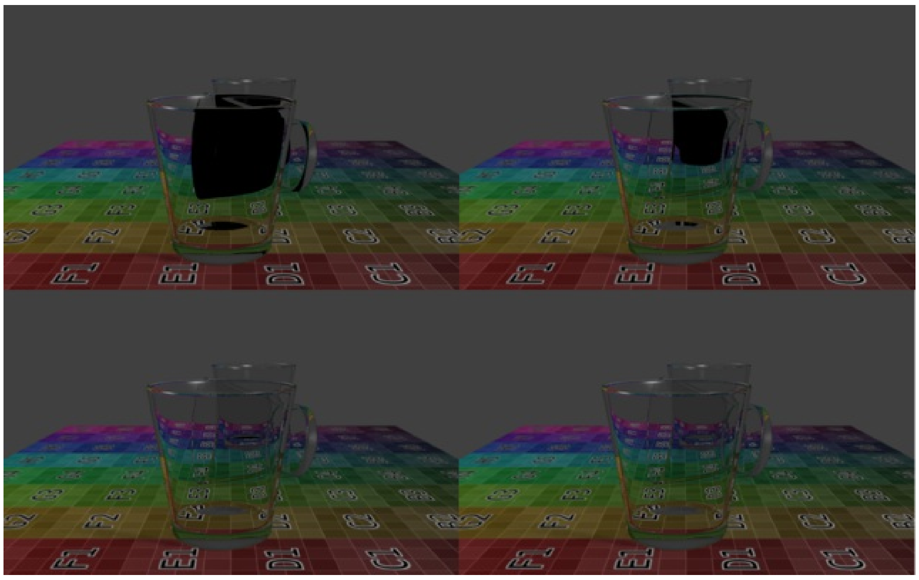
Alpha: 1.000
Fresnel: 2.000
Specular: 1.000
Blend: 2.000

Settings for Fresnel using ray-traced (left) and Z transparency (right).

Note the specular highlight in the F4 glass tile (which is facing midway between the light and the camera); the Fresnel effect can be seen in row C and column 6 where the faces are turned away from the camera.

The amount of Fresnel effect can be controlled by either increasing the Blend value or decreasing the Alpha value.

Depth



A simple scene with three glasses on a surface, and three lamps. Depth was set to 4, 8, 12, and 14, resulting in render times of 24 sec, 34 sec, 6 min, and 11 min respectively.
(Download .blend.)

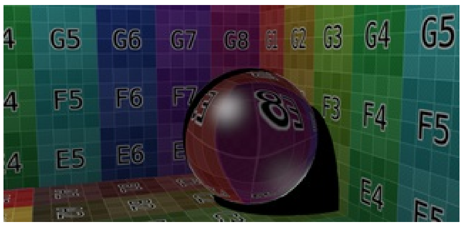
Increasing Depth also considerably increases render time. Each time a light ray passes through a surface, the ray-tracing algorithm is called recursively. In the example above, each side of each glass has an exterior and an interior surface. Light rays thus have to pass through four surfaces for each glass.

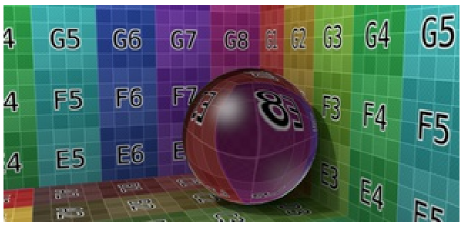
But not only that, at every point on a surface, some of the light can be reflected, or mirrored off the surface in various directions. This results in multiple rays needing to be calculated for each point (often referred to as a **tree of rays**[1]). In each of the rendered images above there are 640×400=256 000 pixels. By increasing Depth, at least one tree of rays is added to each pixel.

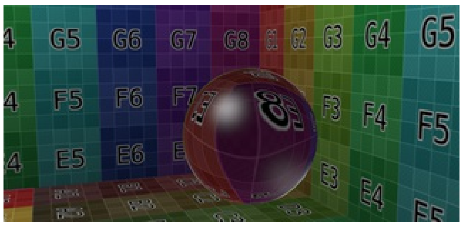
Be kind to your computer. Carefully placing objects in a scene to avoid overlapping transparent objects is often an interesting alternative.

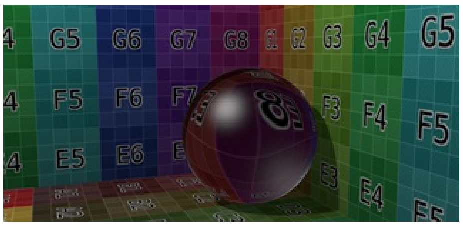
Hints

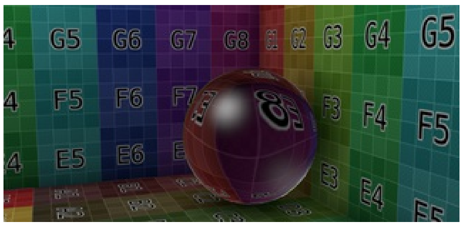
Transparent shadows

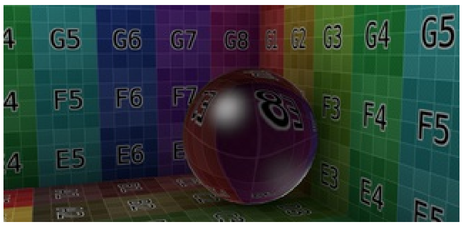

No transparent shadows


No transparent shadows, environment lighting enabled


Transparent shadows enabled, alpha set to 0.0


As previous, alpha set to 0.25


Transparent shadows with ambient occlusion set to multiply, distance 1 (radius of sphere)


As previous, distance increased to 2 (diameter of sphere)

By default, the shadows of transparent objects are rendered solid black, as if the object was not transparent at all. But in reality, the more transparent an object is, the lighter its shadow will be.

In Blender, transparent shadows are set on the materials that receive the shadows from the transparent object. This is enabled and disabled with the Receive Transparent button, in the Material context → Shadow panel. The shadow's brightness is dependent on the Alpha value of the shadow casting material.

Alternatives to transparent ray-traced shadows can be found in the World context, namely the Ambient Occlusion, Environment Lighting, and Gather panels. Alternatively, a texture can be used to control the Intensity value of the shadow-receiving material.

IOR values for Common Materials

The following list provides some index of refraction values to use when ray-traced transparency is used for various liquids, solids (gems), and gases:

A	
Acetone	1.36
Actinolite	1.618
Agalmatolite	1.550
Agate	1.544
Agate	1.540
Air	1.000
Alcohol	1.329
Alcohol, Ethyl (grain)	1.36
Alexandrite	1.745
Alexandrite	1.750
Almandine	1.83
Aluminum	1.44
Amber	1.545

E	
Ebonite	1.66
Ekanite	1.600
Elaeolite	1.532
Emerald	1.560 - 1.605
Emerald Catseye	1.560 - 1.605
Emerald, Synth flux	1.561
Emerald, Synth hydro	1.568
Enstatite	1.663
Epidote	1.733
Ethanol	1.36
Ethyl Alcohol	1.36
Euclase	1.652

J	
Jade, Jadeite	1.64 - 1.667
Jade, Nephrite	1.600 - 1.641
Jadeite	1.665
Jasper	1.540
Jet	1.660
K	
Kornerupine	1.665
Kunzite	1.660 - 1.676
Kyanite	1.715
L	
Labradorite	1.560 -

S	
Sanidine	1.522
Sapphire	1.757 - 1.779
Sapphire, Star	1.760 - 1.773
Scapolite	1.540
Scapolite, Yellow	1.555
Scheelite	1.920
Selenium, Amorphous	2.92
Serpentine	1.560
Shampoo	1.362
Shell	1.530
Silicon	4.24

Amblygonite	1.611
Amethyst	1.540
Ammolite	1.600
Anatase	2.490
Andalusite	1.640
Anhydrite	1.571
Apatite	1.632
Apophyllite	1.536
Aquamarine	1.575
Aragonite	1.530
Argon	1.000281
Asphalt	1.635
Axinite	1.674 - 1.704
Axinite	1.675
Azurite	1.730
B	
Barite	1.636
Barytocalcite	1.684
Beer	1.345
Benitoite	1.757
Benzene	1.501
Beryl	1.57 - 1.60
Beryl, Red	1.570 - 1.598
Beryllonite	1.553
Brazilianite	1.603
Bromine (liq)	1.661
Bronze	1.18
Brownite	1.567
C	
Calcite	1.486
Calspar	1.486
Cancrinite	1.491
Carbon Dioxide (gas)	1.000449
Carbon Disulfide	1.628
Carbon Tetrachloride	1.460
Carbonated Beverages	1.34 - 1.356
Cassiterite	1.997
Celestite	1.622
Cerussite	1.804
Ceylonite	1.770
Chalcedony	1.544 - 1.553
Chalk	1.510
Chalybite	1.630
Chlorine (gas)	1.000768
Chlorine (liq)	1.385
Chrome Green	2.4
Chrome Red	2.42
Chrome Tourmaline	1.61 - 1.64
Chrome Yellow	2.31
Chromium	2.97
Chrysoberyl	1.745
Chrysoberyl, Cat's eye	1.746 - 1.755
Chrysocolla	1.500
Chrysoprase	1.534
Citrine	1.532 - 1.554
Citrine	1.550

F	
Fabulite	2.409
Feldspar, Adventurine	1.532
Feldspar, Albite	1.525
Feldspar, Amazonite	1.525
Feldspar, Labradorite	1.565
Feldspar, Microcline	1.525
Feldspar, Oligoclase	1.539
Flourite	1.434
Formica	1.47
G	
Garnet, Andradite	1.88 - 1.94
Garnet, Demantoid	1.880 - 1.9
Garnet, Demantoid	1.880
Garnet, Grossular	1.738
Garnet, Hessonite	1.745
Garnet, Mandarin	1.790 - 1.8
Garnet, Pyrope	1.73 - 1.76
Garnet, Rhodolite	1.740 - 1.770
Garnet, Rhodolite	1.760
Garnet, Spessartite	1.810
Garnet, Tsavorite	1.739 - 1.744
Garnet, Uvarovite	1.74 - 1.87
Gaylussite	1.517
Glass	1.51714
Glass, Albite	1.4890
Glass, Crown	1.520
Glass, Crown, Zinc	1.517
Glass, Flint, Dense	1.66
Glass, Flint, Heaviest	1.89
Glass, Flint, Heavy	1.65548
Glass, Flint, Lanthanum	1.80
Glass, Flint, Light	1.58038
Glass, Flint, Medium	1.62725
Glycerine	1.473
Gold	0.47
H	
Hambergite	1.559
Hauyne	1.490 - 1.505
Hauynite	1.502
Helium	1.000036
Hematite	2.940
Hemimorphite	1.614
Hiddenite	1.655
Honey, 13% water content	1.504
Honey, 17% water content	1.494
Honey, 21% water content	1.484
Howlite	1.586
Hydrogen (gas)	1.000140
Hydrogen (liq)	1.0974
Hypersthene	1.670
I	
Ice	1.309
Idocrase	1.713
Iodine Crystal	3.34
Iolite	1.522 -

	1.572
Lapis Gem	1.500
Lapis Lazuli	1.50 - 1.55
Lazulite	1.615
Lead	2.01
Leucite	1.509
M	
Magnesite	1.515
Malachite	1.655
Meerschaum	1.530
Mercury (liq)	1.62
Methanol	1.329
Milk	1.35
Moldavite	1.500
Moonstone	1.518 - 1.526
Moonstone, Adularia	1.525
Moonstone, Albite	1.535
Morganite	1.585 - 1.594
N	
Natrolite	1.480
Nephrite	1.600
Nitrogen (gas)	1.000297
Nitrogen (liq)	1.2053
Nylon	1.53
O	
Obsidian	1.489
Oil of Wintergreen	1.536
Oil, Clove	1.535
Oil, Lemon	1.481
Oil, Neroli	1.482
Oil, Orange	1.473
Oil, Safflower	1.466
Oil, vegetable (50° C)	1.47
Olivine	1.670
Onyx	1.486
Opal, Black	1.440 - 1.460
Opal, Fire	1.430 - 1.460
Opal, White	1.440 - 1.460
Oregon Sunstone	1.560 - 1.572
Oxygen (gas)	1.000276
Oxygen (liq)	1.221
P	
Padparadja	1.760 - 1.773
Painite	1.787
Pearl	1.530
Periclase	1.740
Peridot	1.635 - 1.690
Peristerite	1.525
Petalite	1.502
Phenakite	1.650
Phosgenite	2.117
Plastic	1.460
Plexiglas	1.50
Polystyrene	1.55

Sillimanite	1.658
Silver	0.18
Sinhalite	1.699
Smaragdite	1.608
Smithsonite	1.621
Sodalite	1.483
Sodium Chloride	1.544
Spessartite	1.79 - 1.81
Sphalerite	2.368
Sphene	1.885
Spinel	1.712 - 1.717
Spinel, Blue	1.712 - 1.747
Spinel, Red	1.708 - 1.735
Spodumene	1.650
Star Ruby	1.76 - 1.773
Staurolite	1.739
Steatite	1.539
Steel	2.50
Stichtite	1.520
Strontium Titanate	2.410
Styrofoam	1.595
Sugar Solution 30%	1.38
Sugar Solution 80%	1.49
Sulphur	1.960
Synthetic Spinel	1.730
T	
Taaffeite	1.720
Tantalite	2.240
Tanzanite	1.690-1.7
Teflon	1.35
Thomsonite	1.530
Tiger eye	1.544
Topaz	1.607 - 1.627
Topaz, Blue	1.610
Topaz, Imperial	1.605 - 1.640
Topaz, Pink	1.620
Topaz, White	1.630
Topaz, Yellow	1.620
Tourmaline	1.603 - 1.655
Tourmaline	1.624
Tourmaline, Blue	1.61 - 1.64
Tourmaline, Catseye	1.61 - 1.64
Tourmaline, Green	1.61 - 1.64
Tourmaline, Paraiba	1.61 - 1.65
Tourmaline, Red	1.61 - 1.64
Tremolite	1.600
Tugtupite	1.496
Turpentine	1.472
Turquoise	1.610
U	
Ulexite	1.490
Uvarovite	1.870
V-W	
Wardite	1.590
Variscite	1.550
Water (0° C)	1.33346
Water (100° C)	1.31766
Water (20° C)	1.33283

Clinohumite	1.625 - 1.675
Clinozoisite	1.724
Cobalt Blue	1.74
Cobalt Green	1.97
Cobalt Violet	1.71
Colemanite	1.586
Copper	1.10
Copper Oxide	2.705
Coral	1.486
Coral	1.486 - 1.658
Cordierite	1.540
Corundum	1.766
Cranberry Juice (25%)	1.351
Crocoite	2.310
Crystal	2.000
Cuprite	2.850
D	
Danburite	1.627 - 1.641
Danburite	1.633
Diamond	2.417
Diopside	1.680
Dolomite	1.503
Dumortierite	1.686
	1.578
Iron	1.51
Ivory	1.540
Prase	1.540
Prasiolite	1.540
Prehnite	1.610
Proustite	2.790
Purpurite	1.840
Pyrite	1.810
Pyrope	1.740
Q	
Quartz	1.544 - 1.553
Quartz, Fused	1.45843
R	
Rhodizite	1.690
Rhodochrisite	1.600
Rhodonite	1.735
Rock Salt	1.544
Rubber, Natural	1.5191
Ruby	1.757 - 1.779
Rum, White	1.361
Rutile	2.62
Water (gas)	1.000261
Water (35° C, room temp)	1.33157
Whisky	1.356
Willemite	1.690
Witherite	1.532
Vivianite	1.580
Vodka	1.363
Wulfenite	2.300
Z	
Zincite	2.010
Zircon	1.777 - 1.987
Zircon, High	1.960
Zircon, Low	1.800
Zirconia, Cubic	2.173 - 2.21

Mirror Reflections

Mirror reflections are computed in the Blender Render and Cycles render engines using ray tracing. (NB: Reflections are not available in the Game Engine.) Ray tracing can be used to make a material reflect its surroundings, like a mirror. The principle of ray-traced reflections is very simple: a ray is fired from the camera and travels through the scene until it encounters an object. If the first object hit by the ray is not reflective, then the ray takes the color of the object. If the object is reflective, then the ray bounces from its current location and travels up to another object, and so on, until a non-reflective object is finally met and gives the whole chain of rays its color.

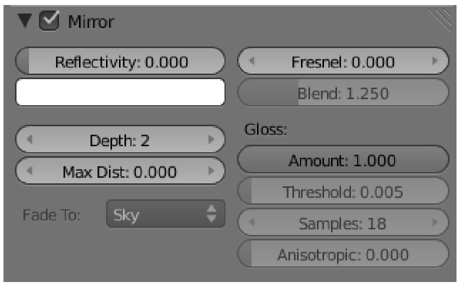
Eventually, the first reflective object inherits the colors of its environment, proportional to its Reflectivity value. Obviously, if there are only reflective objects in the scene, then the render could last forever. This is why a mechanism for limiting the travel of a single ray is set through the Depth value: this parameter sets the maximum number of bounces allowed for a single ray.

Note

You need to enable ray tracing in your scene settings if you want to use ray-traced reflections. This is done in the Scene/Render context → Render Panel. Ray tracing is enabled by default in Blender 2.37 and higher.

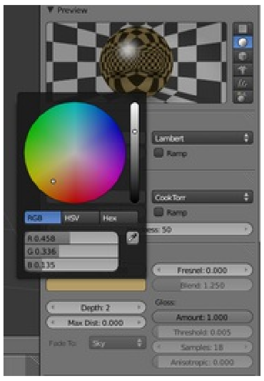
The *Color Swatch* in the mirror panel is the color of the light reflected back. Usually, for normal mirrors, use white. However, some mirrors color the reflection (e.g. metals), so you can change the color by clicking on the swatch. The amount of mirrored reflection is determined by the Reflectivity value. If set to something greater than 0, mirrored reflectivity will be activated and the reflection will be tinted the color set in the swatch.

Options


Mirror
Reflectivity: 0.000
Fresnel: 0.000
Blend: 1.250
Depth: 2
Max Dist: 0.000
Gloss:
Amount: 1.000
Threshold: 0.005
Fade To: Sky
Samples: 18
Anisotropic: 0.000

Enable ray-traced reflections
: Enable or disable ray-traced reflections

Reflectivity
: Sets the amount of reflectiveness of the object. Use a value of 1.0 if you need a perfect mirror, or set it to 0.0 if you don't want any reflection.


Picking a mirror color

Color swatch
: Color of mirrored reflection
: By default, an almost perfectly reflective material like chrome, or a mirror object, will reflect the exact colors of its surrounding. But some other equally reflective materials tint the reflections with their own color. This is the case for well-polished copper and gold, for example. In order to replicate this within Blender, you have to set the Mirror Color accordingly. To set a mirror color, simply click the color swatch in the mirror panel and select a color.

Fresnel
: Sets the power of the Fresnel effect. The Fresnel effect controls how reflective the material is, depending on the angle between the surface normal and the viewing direction. Typically, the larger the angle, the more reflective a material becomes (this generally occurs on the outline of objects).

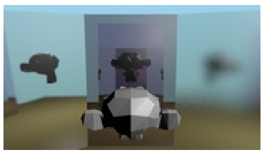
Blend
: A controlling factor to adjust how the blending happens between the reflective and non-reflective areas.

Depth
: Maximum allowed number of light inter-reflections. If your scene contains many reflective objects and/or if the camera zooms in on such a reflective object, you will need to increase this value if you want to see surrounding reflections in the reflection of the reflected object (!). In this case, a Depth of 4 or 5 is typically a good value.

Max Dist
: Maximum distance of reflected rays away from camera (Z-Depth) in Blender units. Reflections further than this range fade out to

reduce compute time.

Fade to
: The color that rays with no intersection within the Max Distance take. Material color can be best for indoor scenes, Sky color (World settings) for outdoor scenes.

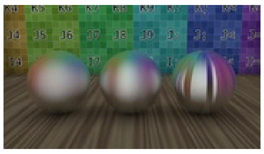

Suzanne in the Fun House (.blend)

Gloss
: In paint, a high-gloss finish is very smooth and shiny. A flat, or low gloss, finish disperses the light and gives a very blurry reflection. Also, uneven or waxed-but-grainy surfaces (such as car paint) are not perfect and therefore slightly need a Gloss < 1.0. In the example to the right, the left mirror has a Gloss of 0.98, the middle is Gloss = 1.0, and the right one has Gloss of 0.90. Use this setting to make a realistic reflection, all the way up to a completely foggy mirror. You can also use this value to mimic depth of field in mirrors.

  Amount
  : The shininess of the reflection. Values < 1.0 give diffuse, blurry reflections and activate the settings below.

  Threshold
  : Threshold for adaptive sampling. If a sampling contributes less than this amount (as percentage), sampling is stopped. Raising the threshold will make the adaptive sampler skip more often, however the reflections could become noisier.

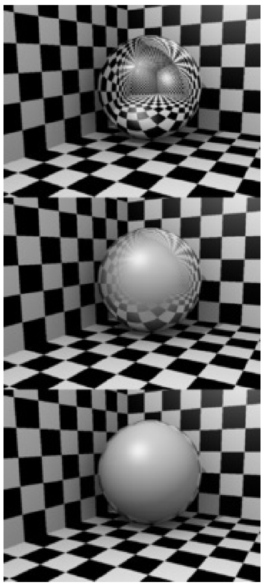
  Samples
  : Number of cone samples averaged for blurry reflection. More samples will give a smoother result, but will also increase render time.


Anisotropic tangent reflecting spheres with anisotropic set to 0.0, 0.75, 1.0. (.blend)

  Anisotropic
  : The shape of the reflection, from 0.0 (circular) to 1.0 (fully stretched along the tangent). If the Tangent Shading is on, Blender automatically renders blurry reflections as anisotropic reflections.
  When Tangent is switched on, the *Anisotropic* slider controls the strength of this anisotropic reflection, with a range of 1.0 (default) being fully anisotropic and 0.0 being fully circular, as is when tangent shading on the material is switched off. Anisotropic ray-traced reflection uses the same tangent vectors as for tangent shading, so you can modify the angle and layout the same way, with the auto-generated tangents, or based on the mesh's UV co-ordinates.

Examples

Fresnel


Demonstration of Fresnel effect with values equal to (from top to bottom) 0.0,

2.5 and 5.0

Let's undertake a small experiment in order to understand what Fresnel is really about. After a rainy day, go out and stand over a puddle of water. You can see the ground through the puddle. If you kneel just in front of the puddle, your face close to the ground, and look again at a distant point on the puddle of water, the liquid surface part which is closer to you lets you see the ground, but if you move your gaze towards the other end of the puddle, then the ground is gradually masked until all you see is the reflection of the sky. This is the Fresnel effect: having a surface sharing reflective and non-reflective properties according to the viewing angle and the surface normal.

In *Demonstration of Fresnel effect with values equal to (from top to bottom) 0.0, 2.5 and 5.0*, this behavior is demonstrated for a perfectly reflective Material (Mirror Reflectivity 1.0).

Fresnel 0.0 stands for a perfect mirror Material, while Fresnel 5.0 could stand for a glossy Material. It's barely noticeable but in the lower picture, the Material is perfectly reflective around the edges.

The smoothness of the Fresnel limit can be further controlled using the Blend slider.

Subsurface Scattering

Many organic and inorganic materials are not totally opaque right at the surface, so light does not just bounce off the top surface. Instead, some light also penetrates the skin surface deeply, and scatters around inside, taking on the color of the insides and emerging back out at a different location. Human/animal skin, the skin of grapes, tomatoes, fruits, wax, gels (like honey, or Jello) and so on all have subsurface scattering (SSS), and photo-realism really cannot be achieved without it.

It is important to understand that subsurface scattering and diffuse are one and the same. The difference is in how far light can diffuse beneath the surface before it is absorbed or transmitted back out.

How it works

Actually calculating the light path beneath the surface of an object is not practical. But it has been shown that it is not necessary to do this, and that one can use a different approach.

Blender calculates SSS in two steps:

- At first the irradiance, or brightness, of the surface is calculated, from the front side of the object as well as from its back side. This is pretty much the same as in a normal render. Ambient Occlusion, Radiosity, the type of diffuse Shader, the light color, etc. are taken into account.
- In the second step, the final image is rendered, but now the SSS shader replaces the diffuse shader. Instead of the lamps, the calculated lightmap is used. The brightness of a surface point is the calculated "Average" of the brightness of its surrounding points. Depending on your settings the whole surface may be taken into account, and it's a bit more complicated than simply calculating the average, but don't bother too much with the math behind it.

Instead let's see what SSS does to a distinct light point.

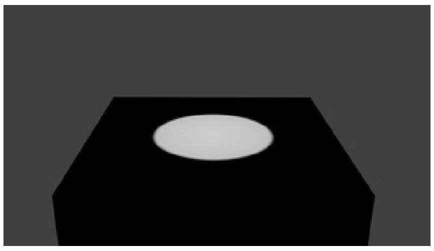


Image 3a: No SSS.

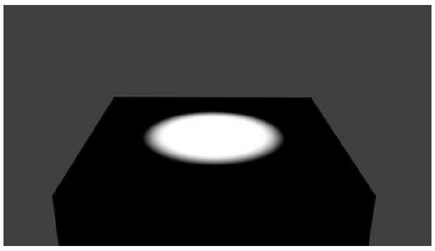


Image 3b: Small SSS radius.

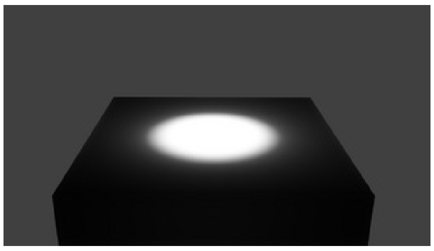


Image 3c: SSS radius enlarged.

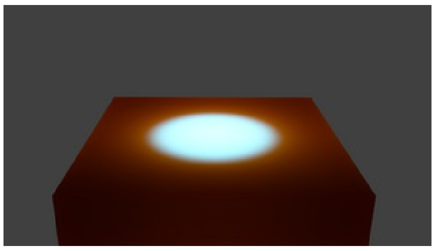


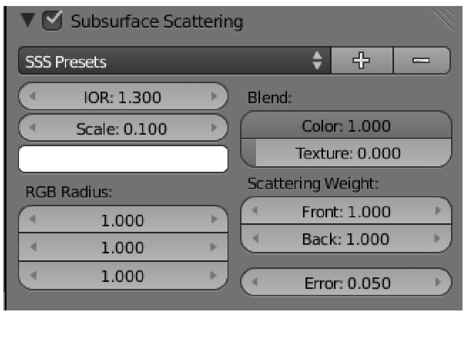
Image 3d: SSS with very large red radius value.

If you turn on SSS, the light is distributed over a larger area. The size of this area depends on the radius values. Instead of distributing all colors with the same amount, you may choose different radius values for each of the RGB colors.

If you use a very large radius value for a color, its light is evenly distributed over the whole object.

Note about scatter radius
Because of the way this scattering is calculated, when using large radius values, you will notice fringing artifacts that appear as the complementary color to the predominant color of the scattering. Above, you see in the last image a bluish band in the illuminated area. This is an unfortunate limitation. A way to lessen this effect is use multiple passes with different scatter radii, and average them.

Enabling Subsurface Scattering


Subsurface Scattering
SSS Presets
IOR: 1.300
Scale: 0.100
RGB Radius:
1.000
1.000
1.000
Blend:
Color: 1.000
Texture: 0.000
Scattering Weight:
Front: 1.000
Back: 1.000
Error: 0.050

- Enable SSS by clicking on the Subsurface Scattering button.
- Accessible at the top are various presets. Add new or remove old presets by clicking the + and - buttons. When you select a preset, the Radius values, the RGB Radius and the IOR are set for you. The remaining options are not set (because they are mostly dependent on the size of your object).

SubSurface Scattering doesn't need ray tracing. But since it is dependent on the incident light and shadows, you need proper shadow calculation (which may need ray tracing).

Options

The numeric sliders control how the light is scattered:

IOR
: The Index Of Refraction value determines the falloff of incident light. Higher values means that light falls off faster. The effect is quite subtle and changes the distribution function only a little bit. By the examination of many different materials, values of **1.3** to **1.5** have been found to work well for most materials. If you know the exact material you are trying to simulate, see our IOR table.

Scale
: The scale of your object, in Blender units, across which you want the scattering effect to take place. Scale 1.0 means **1** Blender unit equals **1** millimeter, scale **0.001** means **1** Blender unit equals **1** meter. If you want to work out what scale value to use in your scene, just use the formula: (size in blender units)/(real world size in millimeters)=scale.

The SSS Color Swatch

Scattering Color (Albedo)
: Albedo is the probability that light will survive a scattering event. If you think of scattering as a filter, this is the height of the filter. It is multiplied by the surface color. In practice, this is unintuitive. It should be the same as the surface color, however changing this value has unintuitive results on the scattering effect:

  1. The darker the color the more light is scattered. A value of 1 will produce no scattering effect.

  So if you set it to green, the lit areas of the object will appear as green, and green is scattered only a little. Therefore the darker areas will appear in red and blue. You can compensate the different scattering by setting a larger radius for the color.

RGB Radius
: This is not in fact the radius of the subsurface scattering, but the average path length between scattering events. As the light travels through the object it bounces around then emerges from the surface at some other point. This value corresponds to the average length the light travels between each bounce. The longer the path length is, the further the light is allowed to scatter. This is the main source of a material's perceived "scatter color." A material like skin will have a higher red radius than green and blue. Subsurface scattering is the diffusion of light beneath the surface. You control how far the light spreads to achieve a specific result.

Blend

: Color
  : This controls how much the R, G, B option modulates the diffuse color and textures. Note that even with this option set to **0.0**, the R, G, B option still influences the scattering behavior.

  Texture
  : How much the surface texture is blurred along with the shading.

Scattering Weight

: Front
  : Factor to increase or decrease the front scattering. When light enters through the front of the object, how much is absorbed or added? (Normally **1.0** or **100%**).

  Back
  : Factor to increase or decrease the back scattering. Light hitting an object from behind can go all the way through the object and come out on the front of the object. This happens mostly on thin objects, like hands and ears.

Error
: This parameter controls how precisely the algorithm samples the surrounding points. Leaving it at **0.05** should give images without artifacts. It can be set higher to speed up rendering, potentially with errors. Setting it at **1.0** is a good way to quickly get a preview of the look, with errors.

Developing your own SSS material

The Traditional Approach

A more common but less intuitive approach is to use "layering". This is a simplified version of the layering approach. See the external links for more information:

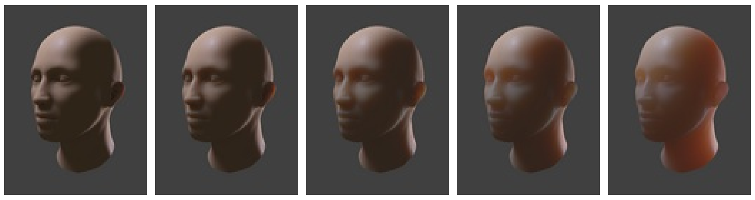
1. Set the SSS color on a value of your choice, normally the predominant color of the object. If you want to use different radii for the colors, don't make it too dark.
2. Set the scale factor. If you want to see much translucency you need small objects or large scale values.
3. Set the radius values.
4. Adjust the brightness with the Front and Back values.

=A more intuitive approach

1. Set the Scattering color to .5
2. Set the Front weight to 2.
3. Set the scale factor based on the size of your object relative to the scene. If you want to see much translucency you need small objects or large scale values.
4. Set the radius values appropriately.

Examples

Skin



Increasing SSS scale (.blend)

See also

- Development Release Log: Subsurface Scattering
- Ben Simonds: *Three Layer SSS in Blender Demystified*

Strands

The Strand section of the Material editor is specific to the rendering of Hair particles. There are two different strand methods available:

- **Polygon strands:** This is the default (old) method. The strands are rendered as flat polygons. The number of polygons depend on the Steps settings in the Render panel of the Object context, Particles sub-context.
- **Strand Primitive:** You activate Strand Primitive with the button Strand render in the Render panel of the particle system. The hair curves are not stored as polygons; only the key points are stored, which are then converted to polygons on the fly. A second difference is the way transparency works. Rather than rendering using the existing system, all strand segments in a part are sorted front to back and rendered in that order.

Strand Primitives

- Are more memory efficient and faster, to make rendering of large amounts of fur and grass possible. For good performance, the render steps button should be lowered (e.g. 2 should be good enough fur), since the result will be a smoothed curve anyway. You need 1 to 2 render steps less than steps in the 3D window. Also, using more render parts helps to reduce memory usage.
- Have a distance of vision reduction (in the Render panel under Child Simplification) for children from faces.
- May be faded out towards the tip without an additional texture.
- Are not ray traced. So they are not visible through ray-transparent materials or in a ray mirror (you can use *Environment Mapping* for that).
- Have shape problems if they are rendered with a greater width.
- Cannot carry a UV-Texture along the strand.

Polygon strands

- Work well with greater width, so you can use them as an alternative to billboards because the strands may have an animated shape.
- Can be textured with a UV-Texture along the strands.
- Are seen by ray tracing.

Strands Shading

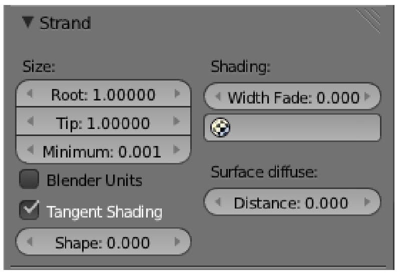


Image 1: Strands Panel.

Strands are rendered with the material of the underlying face/vertex, including shading with a UV-Texture. Since you can assign more than one material to each face, each particle system may have its own material and the material of the underlying face can be different from the material of the strands.

Additionally strands can be shaded along the strand (from root to tip) with a mono-dimensional texture; only polygon strands can carry a two-dimensional UV-Texture.

The options for strand shading are in the Strands section of the Material context.

Root
: Width of the hair at the root.

Tip
: Width of the hair at the tip.

  Minimum
  : This is the minimum thickness (in pixels) of the strands. Strands below that size are not rendered smaller, but are faded to alpha (well, the fading works only for strand primitives). This gives a much better rendering result for thin hair.

Blender Units
: Normally strands are quite thin; the thickness is given in screenpixels. If you use Blender units (BU) you may set the root value up to 2 BU, and the tip value up to 1 BU. You have to consider the overall object size, because the smallest possible size is 0.001 BU. So if you use 1 BU for 1 meter the smallest possible size would be 1 mm (too thick for thin hair).

Use Tangent Shading
: Calculates the light as if the strands were very thin and round. This makes the hair appear brighter and shinier. Disabling the “Tangent Shading” option will still render nicely, but resembles more solid strands, as though made of metal or wood.

Shape
: This slider allows you to control the interpolation. Default (0.0) is a linear interpolation between Root and Tip. A negative value will make the strand narrower (spiky), a positive value will make it fatter.

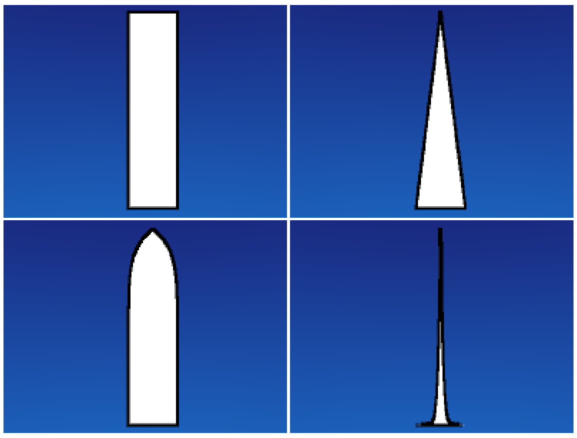


Image 2: a) Root=Tip, b) Tip=0.0, Shape=0.0, c) Shape=0.9, d) Shape=-0.9.

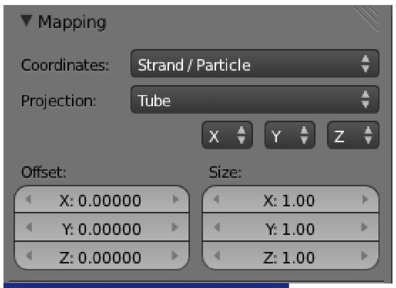
Width Fade
: To fade out along the width of the strand. This works only for Strand Primitives. 0.0 is no fading at all, 1.0 linear fading out.

UV Layer
: You can texture polygon strands with a UV-Texture. Fill in the name of the UV-Set (not the texture) here. You also have to load the texture in the Shading context, Texture and Material sub-contexts (Mapping: UV; you may use every Influence setting you like - especially the alpha value; see *Image 3*).

Surface Diffuse
: Computes the strand normal, taking the normal at the surface into account. This eases the coloring and lighting of hair a lot, especially for Strand Primitives. Essentially hair reacts similar to ordinary surfaces and don't show exaggerated strong and large specular highlights.

  Distance
  : The distance in Blender units over which to blend in the normal at the surface (if you want to use Surface Diffuse only for Grass/Fur at greater distances).

Texturing along the Strand


▼ Mapping

Coordinates: Strand / Particle

Projection: Tube

X Y Z

Offset: X: 0.00000, Y: 0.00000, Z: 0.00000

Size: X: 1.00, Y: 1.00, Z: 1.00

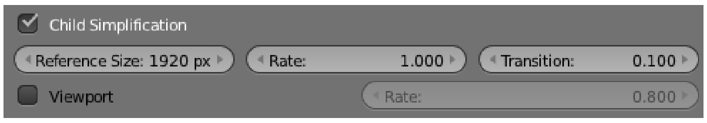
Image 5: …And the render result.

Strands can be textured along the strand, i.e. from root to tip. To do that you have to select Strand/Particle in the Coordinates drop-down in the Mapping panel of the Material sub-context.

Pretty much the most important setting is shown in (*Image 4*), how to fade the tip of a strand to alpha to make nice, fuzzy-looking hair. Normally you would use a linear blend texture for this.

You may of course set any attribute you like, especially color. Be careful with specularity; hairs tend to get too shiny.

Strand render Simplification


✓ Child Simplification

Reference Size: 1920 px | Rate: 1.000 | Transition: 0.100

Viewport | Rate: 0.800

Image 5: Strand render child simplification.

If you use Strand Primitives (Strand render button) and have activated Interpolated Children, the Child Simplification option appears. The strand render has options to remove child strands as the object’s faces become smaller.

Reference Size
: This is the approximate size of the object on screen (in pixels), after which simplification starts.

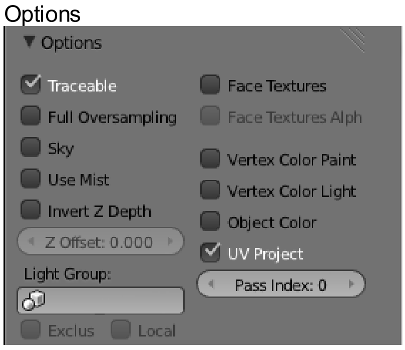
Rate
: How fast strands are removed.

Transition
: The transition period for fading out strands as they are removed.

Viewport
: This removes strands on faces that are outside of the viewport.

  Rate
  : Controls how fast these are removed.

Options



This panel provides a series of control options concerning how objects using this material will appear in the rendered image. All controls are default "Off" unless otherwise stated.

Traceable (default On)
: Include this material and the geometry that uses it in ray-tracing calculations. See Transparency for details of ray-tracing.

Full Oversampling
: Force this material to render full shading/textures for all anti-aliasing samples.

Sky
: Render this material with zero alpha, but with sky background in place (scanline only)

Use Mist
: Use mist on this material (see "World Settings" for more details)

Invert Z depth
Render material's faces with an inverted Z buffer (scanline only)

Z Offset
: Give faces an artificial Z offset for Z transparency.

Light Group
: Limit lighting to lamps in this light group.

Exclusive
: Uses the light group exclusively - these lamps are excluded from other scene lighting

Local
: When linked in, uses local light group with the same name.

Face Textures
: Replace object's base color with color from UV map image textures.

Face Textures Alpha
: Replace object's base alpha with alpha from UV map image textures.

Vertex Color Paint
: Replace object's base color with vertex paint colors (multiply with 'texture face' face assigned textures)

Vertex Color Light
: Add vertex paint colors as additional lighting. (This can be used to produce good incandescence effects).

Object Color
: Modulate the result with a per object color

UV Project (default On)
: Use to ensure UV interpolation is correct for camera projections (use with UV project modifier).

Pass Index
: Index number for the IndexMA render pass.

Shadows

The shadows that appear in a scene are affected by a combination of the layout of objects, the shape of the objects, the materials of the objects, and the lighting. In Blender, the Display Mode (Single Texture, Multitexture,or GLSL) also affects the appearance of shadows. See Doc:2.6/Manual/Lighting/Shadows for a more complete description of this subject.

Shadows in 3D mode

To see shadows in 3D (textured) mode, you must have switched to GLSL mode before making any materials. In MultiTexture mode, shadows only appear in the rendered image: none of these can be seen in the preview image.

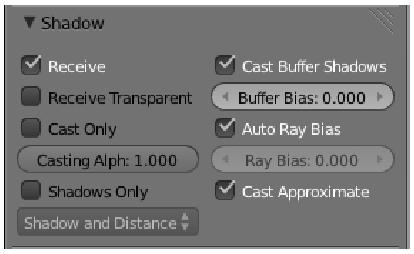

▼ Shadow
- ✓ Receive
- Receive Transparent
- Cast Only
- Casting Alph: 1.000
- Shadows Only
- Shadow and Distance
- ✓ Cast Buffer Shadows
- Buffer Bias: 0.000
- ✓ Auto Ray Bias
- Ray Bias: 0.000
- ✓ Cast Approximate

Fig. 1: Shadow Panel.

The Shadow panel in the Materials Properties window (Fig. 1) controls the effects that the material can have on the shadows that appear in the scene. The various properties are described in the sections below.

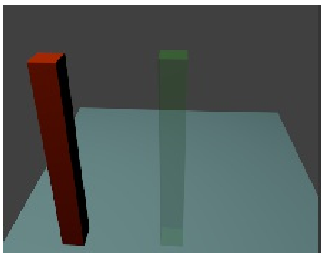


Fig. 2: Scene- all shadow properties off.

Options

The following properties can be set for each individual material with which objects in the scene are shaded. The effects are illustrated with rendered images for a simple scene (Fig. 2) consisting of two "posts", one with a red (totally non-transparent) material; one green (partially transparent) material, set up on a light blue plane to receive the shadows. The illustrations were all taken in Blender Render engine, with Multitexture mode.

Shadow Receiving Object Material

The following options affect the material that receives shadows:

Receive
: Allows this material to receive full-intensity shadows (Fig. 3).

Receive Transparent
: Allows this material to receive shadows whose intensity is modified by the transparency and color of the shadow-casting object (Fig. 4).

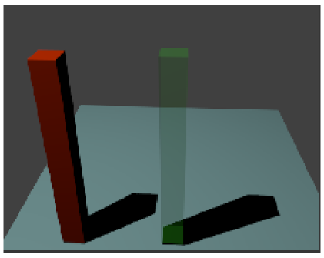


Fig. 3: Plane - Receive.

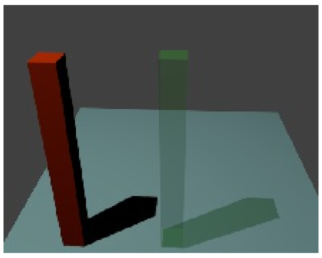


Fig. 4: Plane - Receive + Receive Transparency.

Shadow Casting Object Material

The following options affect the material that casts shadows:

Cast Only
Material appears transparent, but it still casts shadows (Fig. 5).

Casting Alpha
??

Shadows Only
Material appears transparent except for where it receives shadows from other objects, and also it retains its own transparency (Fig. 6). Note the faint image of the partly-transparent post.

Shadow and Distance
???

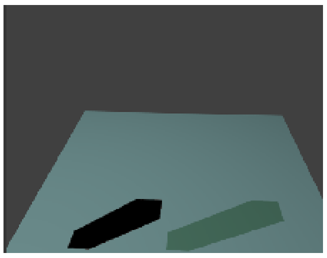


Fig. 5: Posts - Cast Only.

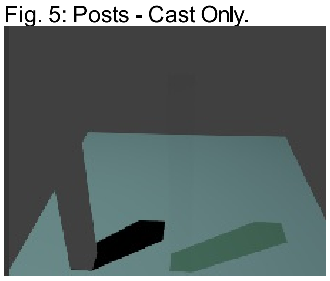


Fig. 6: Posts - Shadows Only.

Buffered Shadow Options

In addition to the shadow options described above, there are further material properties which control buffered shadow features. See section on Spot Buffered Shadows for further discussion of these techniques.

Cast Buffer Shadow
Casts shadows from shadow buffer lamps.

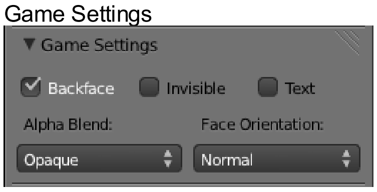
Buffer Bias
Multiplication factor for Buffer shadows (0 = ignore)

Auto Ray Bias -
Prevent raytraced shadow errors on surfaces with smooth shaded normals.

Ray Bias
Bias value to be used.

Cast Approximate
Allow this material to cast shadows when using approximate ambient occlusion.

Game Settings



This panel contains properties that control how the object surfaces that use the material are rendered in real time by the Blender Game Engine.

Game settings are visible when using the game engine for rendering. Material physics usage is described here.

Backface Cull (default On)
: Hide the back faces of objects rendered with this material. If "Off", both sides of the surface are visible (at the expense of lower rendering speed). Note that this setting is applied per-material and not per-face; e.g. if the material is applied to a cube, only the back and front faces of the cube are visible, and not both sides of each face.

Invisible
: Hide all faces of objects rendered with this material.

Text
: Use material as Text object in the Game Engine.

Alpha Blend menu
: Controls how the alpha channel is used to create a transparent texture in the rendered image.

  Alpha Sort
  : Orders the sequence in which transparent objects are drawn on top of each other, so that ones in front receive more light than ones behind.

  Alpha Blend
  : Uses the alpha values present in the bitmap image sourced in the Image slot.

  Alpha Clip
  : Uses the alpha channel as a simple mask.

  Add
  : Render face transparent and add color of face.

  Opaque (default)
  : All alpha values are ignored; the scene is completely non-transparent.

Face Orientation menu

Provides options regarding the orientation (i.e. rotation transformation) of faces to which the material is applied.

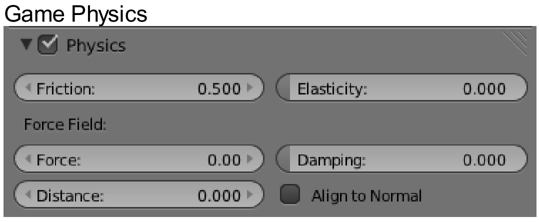
  Shadow
  : Faces are used for shadow.

  Billboard
  : Billboard with Z-axis constraint.

  Halo
  : Screen aligned billboard.

  Normal (default)
  : No transofmation.

Game Physics



Friction: 0.500
Elasticity: 0.000
Force Field:
Force: 0.00
Damping: 0.000
Distance: 0.000
Align to Normal

This panel contains physical properties that control how the object surfaces that use the material are rendered in real time by the Blender Game Engine.

Physics settings are visible when using the game engine for rendering. Game physics usage is described Here

Friction
: Coulomb friction coefficient when inside the physics distance area.

Elasticity
: Elasticity of collisions.

Force Field
: Controls force field settings.

  Force
  : Upward spring force when inside the physics distance area.

  Distance
  : Distance of physics area.

  Damping
  : Damping of the spring force when inside the physics distance area.

  Align to Normal
  : Align dynamic game objects along the surface normal when inside the physics distance area.

Introduction

In addition to creating materials as just described using all the settings on all the materials panels, Blender allows you to create a material by routing basic materials through a set of nodes. Each node performs some operation on the material, changing how it will appear when applied to the mesh, and passes it on to the next node. In this way, very complex material appearances can be achieved.

You should already be familiar with general material concepts and how to create materials/textures using the material menu. You should also have a general understanding of the texture coordinate systems available in Blender (e.g. Generated, UV, etc.). Also, many aspects of a node will be skipped here because in later sections you will see the function expanded upon. Each section builds off the previous.

To start, the node system does not make the material menu obsolete. Many features and material settings are still only accessible through the material panel (e.g. Ray Mirror). However with the advent of nodes, more complex and fantastic materials can be created since we now have greater control.

Just in case you're not (yet) familiar with the concepts: when you create a system of nodes (otherwise known as a "noodle"), you're describing a data-processing pipeline of sorts, where data "flows from" nodes which describe various *sources,* "flows through" nodes which represent various processing and filtering stages, and finally "flows into" nodes which represent outputs or destinations. You can connect the nodes to one another in many different ways, and you can adjust "knobs," or parameters, that control the behavior of each node. This gives you a tremendous amount of creative control. And, it will very quickly become intuitive.

Having said all that, let’s begin with a normal material.

Here we have the standard material we have added to a cube mesh. We could, as we have in the past, add color and other settings to this material and it would certainly look nice. But let’s say we are just not getting what we are looking for? What if we want to control the creation more tightly or add more complexity? Here is where nodes come in.

Making this node map is accomplished by working in a Node Editor window. This section covers:

- Enabling Material Nodes.
- The Node Editor window, its basic controls, and working with nodes.
- The specific types of nodes available for materials.

Nodes Concepts

Nodes

"Nodes" are individual blocks that perform a certain operation, and might have one or many different outputs.

Conceptually, there are three basic types of nodes:

- **Input Nodes**

  these nodes *produce* information, but do not have any inputs of their own.
  Examples are: *Render Layers*, *Value* and *RGB* nodes.

- **Processing Nodes**:

  these nodes *filter* or *transform* their inputs, to produce one or more outputs.
  Examples are: *RGB Curves*, *Defocus,'* and **Vector Blur** nodes.

- **Output Nodes**:

  these nodes *consume* their inputs to produce some kind of meaningful result.
  Examples are: *Composite* node (which determines the final output used by Blender), *Viewer* (which displays the output of a socket), and **File Output** node.

Noodles

The essential idea of nodes is that you can create an arbitrarily-complex *network* of nodes, by connecting the *outputs* of one or more nodes to the *inputs* of one or more other nodes. Then, you can set appropriate parameters (as you see fit) for each node.

This network is called a "noodle" and it describes how information literally *flows through* to produce whatever result you want.

Node Groups

You can define *node groups*, and use those groups as they were a single node.

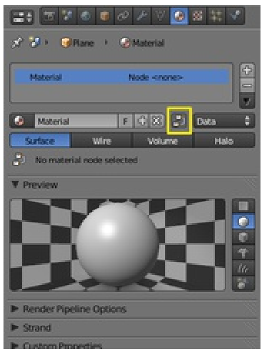
You can link and append these node groups from other files.

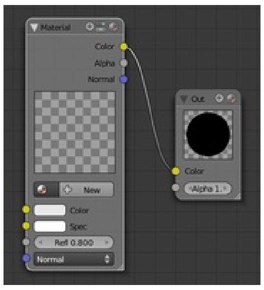
Accessing The Node Editor

First lets enter the node editor and make sure that the node editor has the material node button (the sphere icon) pressed, not the composite or texture node buttons.

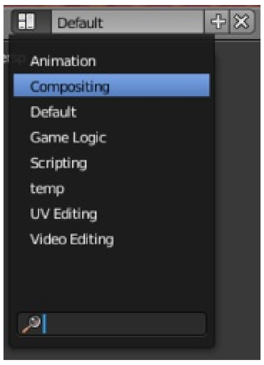
Enabling Node Materials in the Material Buttons

Let’s take the base material and hit the Nodes button next to the material name in the material panel or the node editor. You will see a change in the material panel.


Material's menu with Nodes enabled


Default nodes

What you have just done is told Blender to make the material you were on to become the node tree. Most of the panels we normally find in the material menu are now gone.



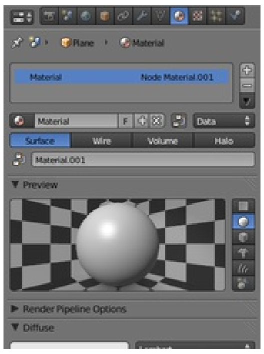
Accessing the Compositing screen

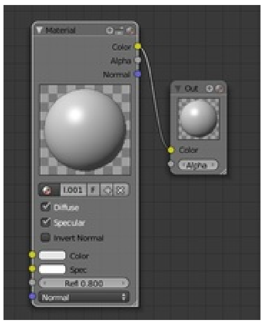
If you switch to the Compositing screen (Ctrl← if you are on the default screen) you'll find a Node Editor on the top half of the screen. When you enabled material nodes, a material node and an output node were automatically added to the node editor.

You can also split the 3D view in the default screen in two and change one into a Node Editor.

It is important to note that you can add a new material (which you can edit and change like any other material in the material panel), add an already created material or append a material from another blender file, and also use the material that you used to create the node tree.

Here, we added a new material in the Node editor (*Material.001*), and as we did, we can access the properties of this material in the material's menu.


Material's menu with a first material added to the Node Editor


A first material added to the noodle

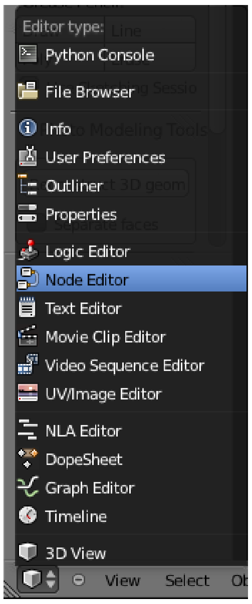
External Links

- Blender Material Nodes - Changelog for the Blender version that introduced material nodes.

The Node Editor

This section explains the window in general, and its header menu options. It also tells you how to enable nodes for use within Blender.

Accessing The Node Editor

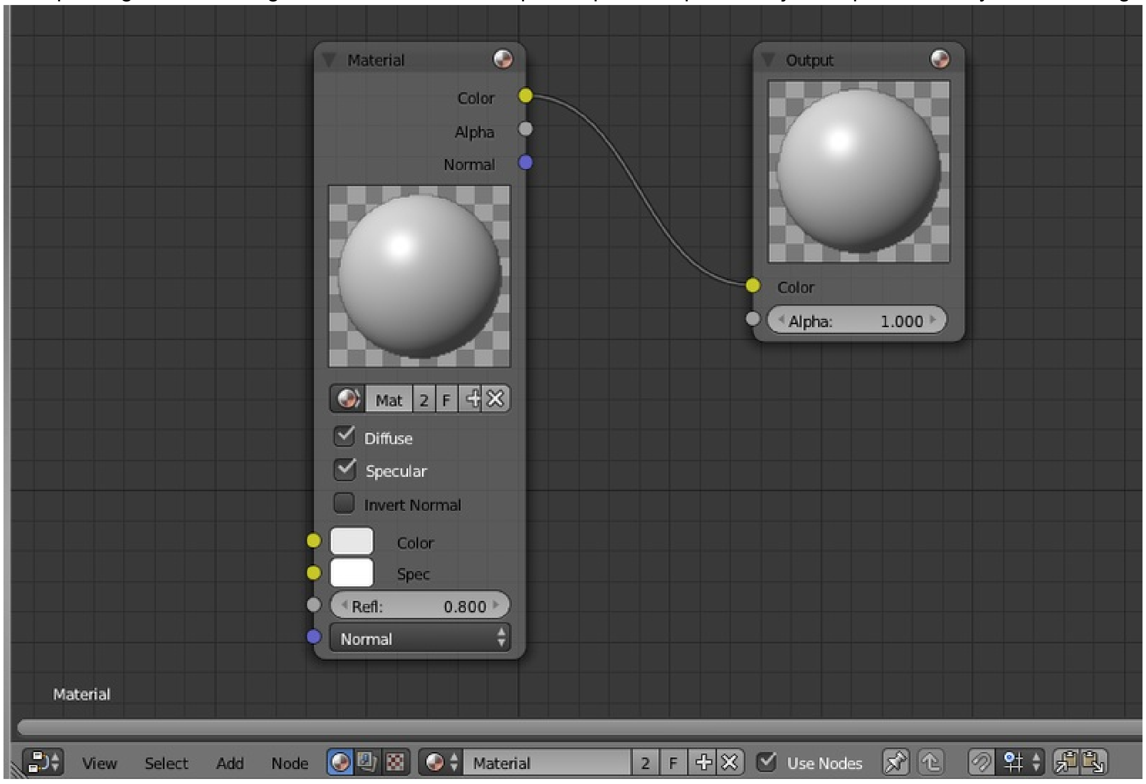


Select the Node Editor window.

First let's enter the node editor by changing our window type to Node Editor. As shown in *Select the Node Editor window*, click on the window type icon and select Node Editor from the popup list. Node maps can get quite large, so use or create a big window. The window has a graph-paper style background and a header.

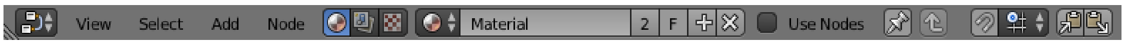
Each scene within your blend file can have multiple Material Node maps and ONE Compositing Node map. The Node Editor window shows either type of map, depending on the selector position.

Hint
You might want to add a new window layout called 6-Nodes (the list is shown on the User Preferences header at the top of your screen) comprised mostly of one big Node Editor window. My layout has the buttons window at the bottom and a text editor window on the side for me to keep notes. If you have a widescreen display (or even a regular one), you might also want to add a 3D view or UV/Image Editor window to the left side of the Node window layout, so you can work with images or your model while you're manipulating nodes. Having the 3D Preview Render panel open on top of an object is quite useful if you're tweaking material nodes.



Node Editor.

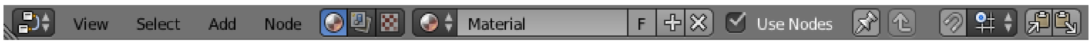
By default, the header, when first displayed, is uninitialized as shown:

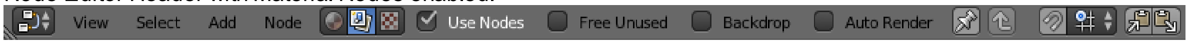


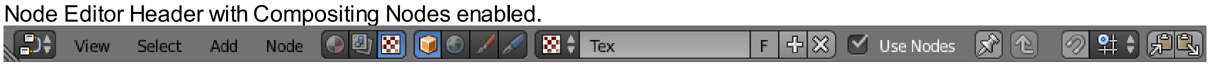
Default Node Editor header.

Activating Nodes

- What nodes to use?
  - If you want to work with a material node map, click the ball in the Material/Compositing node set selector. (See *Node Editor Header with Material Nodes enabled.*)
  - If you want to work with a compositing node map, click the overlaped pictures on the Material/Compositing node set selector. (See *Node Editor Header with Compositing Nodes enabled.*)
  - If you want to work with a texture node map, click the checker on the Material/Compositing node set selector. (See *Node Editor Header with Texture Nodes enabled.*)
- To actually activate nodes, click the Use Nodes button.
- The first time that you select either a Material, Compositing or a Texture node map, the Node Editor window will be instantly filled with starter input and output compositing nodes already connected together.



Node Editor Header with Material Nodes enabled.


Node Editor Header with Compositing Nodes enabled.


Node Editor Header with Texture Nodes enabled.

Node Editor Window Actions

When the cursor is in the window, several standard Blender hotkeys and mouse actions are available, including:

Popup menu
: Space - Brings up a main popup menu, allowing you to add, view, select, etc.

Delete
: X or Del - Deletes the selected node(s).

Box select
: B - Starts the bounding box selection process. Position your cursor and LMB click & drag to select a set of nodes.

Cut connections (lasso)
: CtrlAlt LMB click & drag - Starts a lasso selection, BUT when you let up the mouse button, all threads (connections) within the lasso are broken.

Undo
: CtrlZ Very helpful if you forgot to press B before box-selecting, eh?

Redo
: CtrlY or Ctrl⇧ ShiftZ - You can use this if you used "undo" a bit too often :)

Select multiple
: ⇧ Shift LMB or ⇧ Shift RMB - Multiple node select.

Grab/Move
: G - Moves your current selection around.

Standard Window Control
Node maps can get pretty hairy (large and complicated, that is). The contents of the window (the node map) can be panned just like any other Blender window by clicking MMB and dragging about. Wheeling Wheel up/down or using the keypad + NumPad/- NumPad will zoom in/out. The window can be resized and combined using the standard window techniques (see *Navigating in 3d Space*).

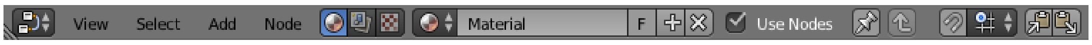
Node Editor Header

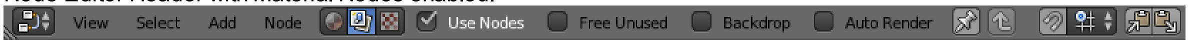
At a glance

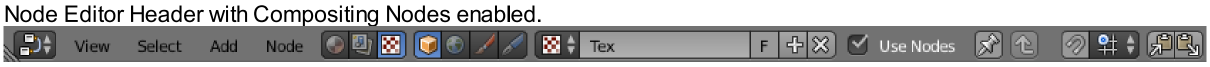
On the window header, you will see header options:

- View - to see things more clearly;
- Select - to do things more clearly;
- Add - to walk with...err..to add Nodes, organized by type;
- Node - to do things with selected nodes, akin to vertices;
- a Material, Compositing or Texture node set selector;
- a Use Nodes button;
- a Use Pinned button;
- a Go to Parent button;
- a Snap button;
- a Snap Node Element selector;

- a Copy Nodes button;
- a Paste Nodes button.



Node Editor Header with Material Nodes enabled.


Node Editor Header with Compositing Nodes enabled.


Node Editor Header with Texture Nodes enabled.

Menus

View, Select and Add

These popup menus provide the basic functions:

View
: This menu changes your view of the window, standing in for the standard keyboard shortcuts + NumPad (zoom in), - NumPad (zoom out), ↖ Home (zoom all) or equivalent mouse actions.

Select
: This menu allows you to select a node or groups of nodes, and does the same as typing the hotkey to select all A or start the border select B process.

Add
: This menu allows you to add nodes. Please see the next section for a discussion on the types of nodes that you can add, and what they do. Clicking this menu item is the same as pressing Space when the cursor is in the window

Node

Hide
: H - Hides your selected nodes. Just like vertices in a mesh.

Grouping
: Most importantly, this menu option allows you to create a user-defined group of nodes. This group can then be edited and added to the map. To create a group, select the nodes you want, and then Node → Make Group, or just use the keyboard shortcut CtrlG. Edit the name using the little input box in the group. Groups are easily identified by their green header and cool names you have picked for them.

Delete
: X - Deletes selected nodes.

Duplicate
: ⇧ ShiftD - Makes an Unlinked copy, with the same settings as the original.

Grab
: G - Moves the little nodes around according to your mouse, just like with meshes.

Duplicate - Faked you out
The new copy is placed **exactly over the old one**. But it isn't the connected one, so playing with the controls will do nothing to your images, even though it **looks** like it's connected with the little threads coming out of the node that is **underneath**. You have to move the duplicated node to reveal the connected node beneath it.

Grab - Reminder Only
Just like my mother-in-law, the menu item does not actually do anything; it's just there to remind you that you can press the G key when your cursor is in the window and actually accomplish something with your life (like rearranging nodes in the window).

Buttons

Material/Composite/Texture Selector

Nodes are grouped into two categories, based on what they operate on:

- to work with Material Nodes, click on the ball,
- to work with Compositing nodes, click on the overlaped pictures,
- to work with Texture nodes, click on the checker.

Use Nodes Button

This button tells the render engine to use the node map in computing the material color or rendering the final image, or not. If not, the map is ignored and the basic render of the material tabs or scene is accomplished.

Use Pinned Button

This button tells the render engine to use pinned node tree.

Go to Parent Button

This button allows you go to parent node tree.

Snap Button

Toggle snap mode for node in the Node Editor window.

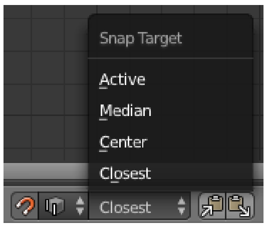
Snap Node Element selector
: This selector provide the follow node elements for snap:

  Grid (default)
  : Snap to grid of the Node Editor window.

  Node X
  : Snap to left/right node border.

  Node Y
  : Snap to top/bottom node border.

  Node X/Y
  : Snap to any node border.
  : Snapping to node border takes into account snap target:



Snap Target.

Snap Target
: Which part to snap onto the target
: Closest: Snap closest point onto target.
: Center: Snap center onto target.
: Median: Snap median onto target.
: Active: Snap active onto target.

Copy Nodes Button

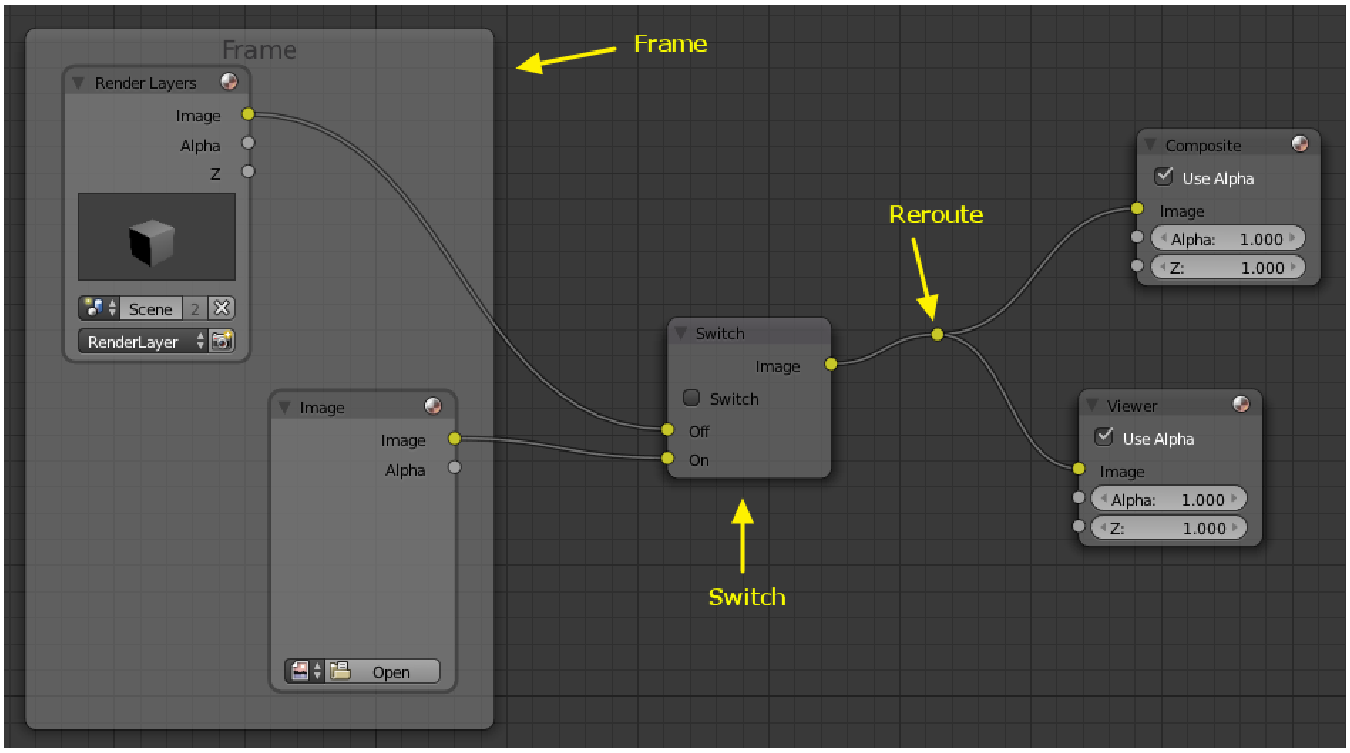
This button allows you copy selected nodes to the clipboard.

Paste Nodes Button

This button allows you paste nodes from the clipboard to the active node tree.

Layout Nodes

Layout nodes are designed for enhanced arrangement your nodes in Node Editor window. They are available from menu Add -> Layout.



Examples of the Layout Nodes.

Blender provides the following layout nodes:

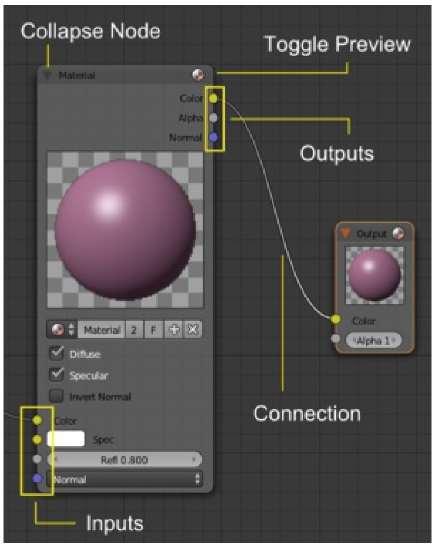
Frame
: This node is used for simple join a several nodes in the single place.

Reroute
: It's intended for branching threads with the purpose of optimization of Node Editor window's space.

Switch
: This node is applied for switching between color values. Available only for Compositing nodes.

Node Controls

This page explains the widgets to control a node.



Nodes main controls

Titlebar
: This contains the node's name, along with several different collapse buttons.

Input sockets
: The left side of a node has input sockets:

- *blue sockets* accept vectors.
- *yellowsockets* accept colors.
- *gray sockets* accept single values (like alpha).

Output sockets
: The right side of a node has output sockets:

- *blue sockets* produce vectors.
- *yellowsockets* produce colors.
- *gray sockets* produce single values (like alpha).

Image preview
: Inside the node there's an area to show the image preview being output by the node or the curves that control the node behavior (for example in a RGB node).

Buttons and menus
: Below the image preview there are buttons and menus to control the node behavior.

Threads
: A curved line shows a connection from an output socket to an input socket. The socket types must match.

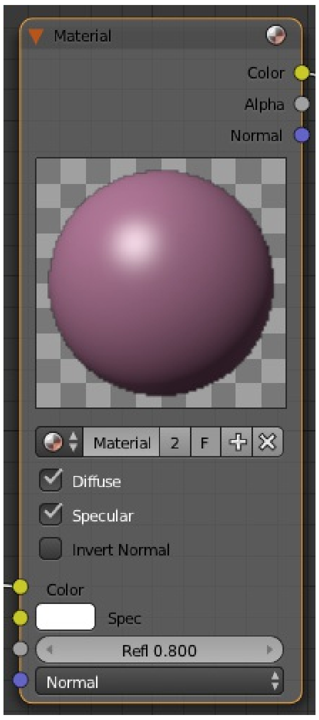
Connections associated with the active node are highlighted for better visibility.

Collapsing toggles

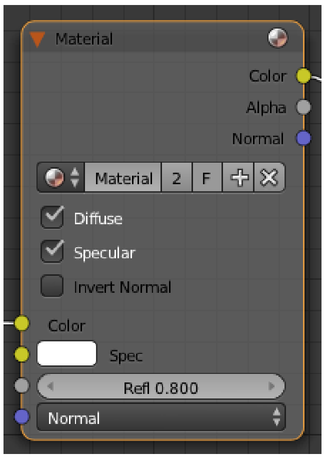
At the top of a node there are up to 4 visual controls for the node (*Top of a Node*). Clicking these controls influences how much information the node shows.

Node toggle (▼ ▶)
: The arrow on the left collapses/uncollapses the node.

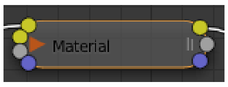
Preview image toggle
: The sphere button on the far right of the titlebar hides/unhides the preview image.



Full display



Preview hidden

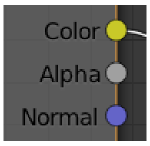


Node collapsed

Sizing the node

Fine Sizing of an individual node can also be accomplished somewhat by clicking LMB and dragging on the left or right edge of the node.

Sockets



Node
Sockets.

Each Node in your node window will have "sockets" (often also referred to as "connectors") which are small colored circles to which input data and output data can be linked (*Node Sockets*).

The sockets on the left side of a node describe *inputs,* while the sockets on the right side are *outputs.*

For your convenience, nodes are *color-coded* according to the type of information they expect to send or receive. There are three colors:

● Yellow sockets
: Indicates that **color** information needs to be input or will be output from the node.

● Gray sockets
: Indicates values (**numeric**) information. It can either be a single numerical value or a so-called "value map." (You can think of a value map as a grayscale-map where the different amount of bright/dark reflects the value for each point.) If a single value is used as an input for a "value map" socket, all points of the map are set to this same value.
Common use: Alpha maps and value options for a node.

● Blue/Purple sockets
: Indicates **vector/coordinate/normal** information.

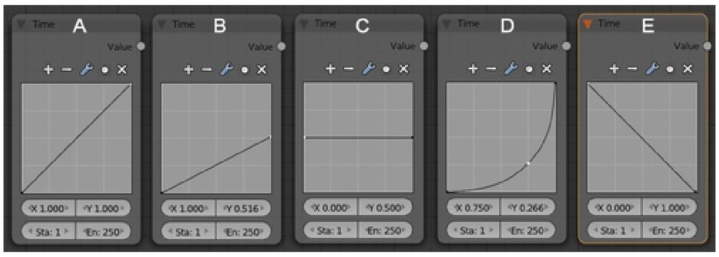
Between nodes, yellow must be linked to yellow, gray to gray, blue to blue, unless you use a *converter,* which we'll cover later on.

Next to the color in the node you will see the name of that socket. Though not always the case, you can think of the name of the socket as what the information is *intended* to be. But this is not necessarily what it *has* to be. For example, I can add a link from a gray socket titled Alpha to the material node's gray Reflection socket and still get a result, the key thing being that it's a "gray to gray" connection.

There are exceptions where you can mix yellow (i.e. a color image) and gray (*e.g.* grayscale) without converters. Blender normally places a converter if needed, so feel free to experiment with them. You can use the "Viewer" output nodes, as explained in the later sections, to see if/how it works.

Curves

Some nodes have a curve area that translates an input value to an output value. You can modify this curve shape by clicking on a control point and moving it, or adding a control point. Some examples are shown below:



Modifying a curve node.

Every curve starts out as a straight line with a slope of 1. The curve starts out with two tiny black control points at each end of the line. Clicking LMB on a control point selects it and it turns white.

Changing the curve affects how the output is generated. The input, X, usually proceeds linearly (at regular intervals) across the **bottom** axis. Go up until you hit the curve, and then over to the **right** to determine the Y output for that corresponding X. So, for the second example, as X goes from 0 to 1.0 across the bottom, Y varies from 0.0 to 0.5. In the third, as X goes from 0.0 to 1.0 across the bottom, Y stays constant at 0.5. So, in the picture above, these curves have the following effect on time: **A** don't affect, **B** slow down, **C** stop, **D** accelerate, and **E** reverse time.

The "Curves" widget is a built-in feature in Blender's UI, and can be used anywhere, provided the curve data itself is being delivered to this widget. Currently it is used in the Node Editor and in the UV Window.

This widget will map an input value horizontally and return the new value as indicated by the height of the curve.

Note: The fact that one of the points on the curve is "white" in each of these screenshots is *not* significant; it just means that it happened to be the point most-recently selected by your author when preparing this tutorial. What matters here is the shape of *the curve,* not the position (nor the color) of the control points that were used to define it.

RGB Curves

Multiple curves can be edited in a single widget. The typical use, RGB curves, has "Combined" result or "Color" ("C") as the first curve, and provides curves for the individual R, G, and B components. All four curves are active together; the "C" curve gets evaluated first.

Selecting curve points

- LMB always selects 1 point and deselects the rest.
- Hold ⇧ Shift while clicking to extend the selection or select fewer points.

Editing curves

- LMB click&drag on a point will move points.
- A LMB click on a curve will add a new point.
- Dragging a point exactly on top of another will merge them.
- Holding ⇧ Shift while dragging snaps to grid units.
- Ctrl LMB adds a point.
- Use the X icon to remove selected points.

Editing the view

The default view is locked to a 0.0-1.0 area. If clipping is set, which is the default, you cannot zoom out or drag the view. Disable clipping with the icon resembling a #.

- LMB click&drag outside of curve moves the view
- Use the + and - icons to zoom in or out.

Special tools

The wrench icon gives a menu with choices to reset a view, to define interpolation of points, or to reset the curve.

Using nodes

Adding Nodes

Nodes are added in two ways to the node editor window:

- By clicking the Add menu in the node editor toolbar and picking the type of node you want, or
- By clicking the ⇧ ShiftA -> Add and picking a node from the popup Add menu.

Arranging Nodes

In general, try to arrange your nodes within the window such that the image flows from left to right, top to bottom. Move a node by clicking on a benign area and dragging it around. The node can be clicked almost anywhere and dragged about; connections will reshape as a bezier curve as best as possible.

Connecting nodes

LMB -click and drag a socket: you will see a branch coming out of it: this is called a "thread".

Kepp dragging and connect the thread to an input socket of another node, then release the LMB .

In this case, a copy of each output is routed along a thread. However, only a single thread can be linked to an input socket.

Disconnecting nodes

To break a link between sockets Ctrl LMB -click in an empty areas near the thread you want to disconnect and drag: you will see a little cutter icon appearing at your mouse pointer. Move it over the thread itself, and release the LMB .

Duplicating a node

Click LMB or RMB on the desidered node, press ⇧ ShiftD and move the mouse away to see the duplicate of the selected node appeaing under the mouse pointer.

Gotcha!
When you duplicate a node, the new node will be positioned *exactly* on top of the node that was duplicated. If you leave it there (and it's quite easy to do so), you can **not** easily tell that there are *two* nodes there! When in doubt, grab a node and move it slightly to see if something's lurking underneath.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in the node editor, making your material or composite 'noodle' (node network) easier to work with. Grouping nodes also creates what are called NodeGroups (inside a .blend file) or NodeTrees (when appending).

Conceptually, "grouping" allows you to specify a *set* of nodes that you can treat as though it were "just one node." You can then re-use it one or more times in this or some other .blend file(s).

As an example: If you have created a material using nodes that you would like to use in another .blend file, you *could* simply append the material from one .blend file to another. However, what if you would like to create a new material, and use a branch from an existing material node network? You could re-create the branch. Or you could append the material to the new .blend file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different .blend files. What if you have created a “Depth of Field” composite node network and would like to use it in another .blend file? What if you wanted to apply exactly the same series of operations dozens of times? Here again, you *could* re-create the network, but this is not very efficient. A better method of re-use, for either material node branches or composite node networks, would be to create groups of nodes.

Once a group has been defined, it becomes an opaque object; a reusable software component. You can (if you choose) ignore exactly how it is *defined,* and simply use it (as many times as you like) for whatever it *does.* Groups can be made available through the Blender library and standard appending method.

Grouping Nodes

Panel: Node Editor

Menu: ⇧ ShiftA → Group → Make Group

To create a node group, in the node editor, select the nodes you want to include, then press CtrlG or ⇧ ShiftA » Group » Make Group. A node group will have a green title bar. All of the selected nodes will now be minimized and contained within the group node. Default naming for the node group is *NodeGroup, NodeGroup.001* etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one .blend file to another, Blender does not make a distinction between material node groups or composite node groups, so I recommend some naming convention that will allow you to easily distinguish between the two types. For example, name your material node branches *Mat_XXX,* and your composite node networks *Cmp_XXX.*

What not to include in your groups (all types of Node editors)

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not include**:

Source nodes
: if you include a source node in your group, you'll end up having the source node appearing *twice:* once inside the group, and once outside the group in the new material node-network.

: Examples of source nodes are: the *Material Node* (Material nodes editor) and the *Render Layers Node* (Composite Editor).

Output node
: if you include an output node in the group, there won't be an output socket available *from* the group!

: Examples of output nodes are: the *Output Node* (Material nodes editor) and the *Viewer Node* (Composite Editor).

Editing Node Groups

With a group node selected, pressing ⇆ Tab expands the node to a window frame, and the individual nodes within it are shown to you. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of your editor window. You will not be able to thread them to an outside node directly from them; you have to use the external sockets on the side of the Group node. To add or remove nodes from the group, you need to ungroup them.

Ungrouping Nodes

The AltG command destroys the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

Appending Node Groups

Once you have appended a NodeTree to your .blend file, you can make use of it in the node editor by pressing ⇧ ShiftA → Add → Group, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Types of Material Nodes

This section is organized by type of node, which are grouped based on similar functions:

- Input - Introduces a material or component to the node map.
- Output - Displays the result in progress as a small image.
- Color- Manipulates the colors of the material.
- Vector- Change the way light is reflected off the material.
- Convertors- Convert colors to other material colors.
- Groups- User-defined groups of nodes.
- Dynamic- Custom nodes defined by Python. These are also known as PyNodes.

Material Input Nodes

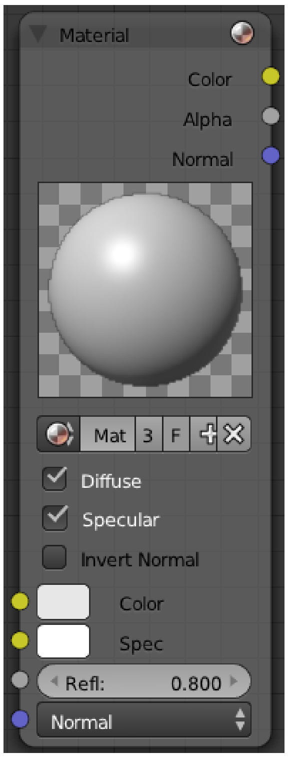
A starting material is created in the Materials Panel. The Nodes button is enabled to add that material to the list of noded materials shown in the Node Editor window header. Other inputs to the node map include:

- A value
- A color
- A texture
- Geometry
- Material
- Camera Data
- Lamp Data

Material Node

Panel: Node Editor → Material Nodes

Menu: ⇧ ShiftA → Input → Material



Material node

The Material node is used to add a material to the node program. Materials can be anything from pure shading to fully layered with textures. It inputs the main attributes of a material (color, alpha and normal vector) into the map.

Output

Materials can output color (which includes shading and any textures assigned to it), alpha, and the final normal calculated from any textures it has.

- Color - value of the color, combined by the node.
- Alpha - value of the alpha, combined by the node.
- Normal - direction of the normal, combined by the node.

Input

Materials can take inputs for colors, inputs for diffuse color and specularity color, a value for reflectivity, and a normal.

- Color - The base color of the paint. Can be set
  - manually by LMB clicking on the color swatch applet next to the socket, choosing a color using the control panel that pops up, and pressing ↵ Enter
  - based on an Active Material which is specified using the material panels, or
  - plugged in from an RGB color generator.
- Spec - The color that is reflected as you get perpendicular to the light source reflecting off the surface. The color can be
  - plugged in from another node or
  - set manually by LMB clicking on and using the color swatch applet.
- Refl: - The degree to which the material reflects light and gives off its color. The value can be provided by another node or set manually.
- Normal - The lighting condition.

Controls

Material field
: You can browse and select materials here.

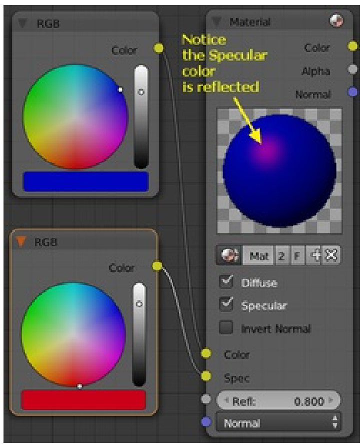
Diffuse toggle
: Turn on/off Diffuse Color.

Specular toggle
: Turns on/off Specularity calculation.

Invert Normal toggle
: Inverts the material input normal when activated (which, of course, is a combination of the 3D normal given to it by the 3D object plus the normal input point).

Normal Override
The normal input socket does not in any way blend the source normal with the underlying geometry. Any plugged in Geometry here overrides the Normal lighting conditions.

Using the Material Node with Specularity



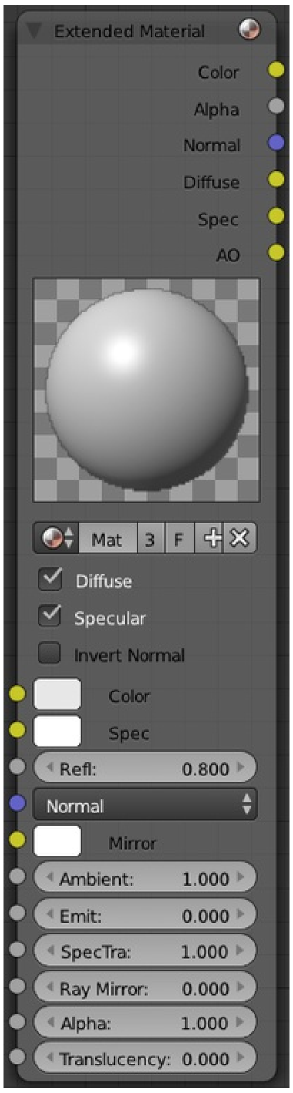
Material Node using Specularity

To make a material node actually generate a color, you have to specify at least a basic input color, and optionally a specularity color. The specularity color is the color that shines under intense light.

For example, consider the mini-map to the right. The base color, a dark blue, is connected from an RGB color generator node to the Color input socket. The specular color, yellow, is connected to the Spec input. Under Normal lighting conditions on a flat surface, this material will produce a deep blue color and, as you approach a spot perpendicular to the light, you will see the yellow specular color mix in.

Enable Spec
To see specularity, you have to enable it by clicking the blue Spec button located just below the material color swatch in the node.

Extended Material Node



Extended Material node

Adds additional input and output channels to the material node.

Input

Color
: Includes a color swatch, allowing you to select the color directly on the node.

Mirror Color
: Color of mirrored reflection.

Ambient
: Amount of global ambient color the material receives.

Emit
: Amount of light to emit.

SpecTra
: Alpha for the specular color.

Ray Mirror
: Amount of reflectiveness of the object.

Alpha
: Transparency of the material by setting all pixels in the alpha channel to the given value.

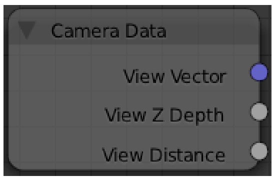
Translucency
: Amount of diffuse shading on the back side

Output

Materials can additionaly output the followings:

- Diffuse - value of the diffuse color, combined by the node.
- Spec - value of the specular color, combined by the node.
- AO - value of the Ambient Occlusion, combined by the node.

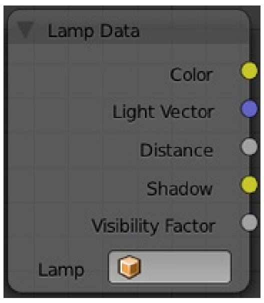
Camera Data Node



Camera Data node

View Vector
    A Camera space vector from the camera to the shading point.
View Z Depth
    How far away each pixel is from the camera
View Distance
    Distance from the camera to the shading point.

Lamp Data Node



Lamp Data node

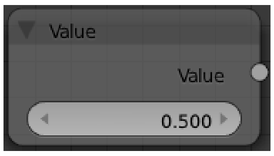
The Lamp Data node is used to obtain information related to a specified lamp object. Select a lamp object listed in the Lamp field, then the following outputs will be available:

Color
    Lamp color multiplied by the lamp energy.
Light Vector
    An unit vector in the direction from the shading point to the lamp.
Distance
    Distance from the shading point to the lamp.
Shadow
    Shadow color that the other objects cast on the shading point.
Visibility Factor
    Light falloff ratio at the shading point.

The light textures and the shadow textures affect the Color and Shadow outputs, respectively.

Portability to Various Scenes
If multiple materials use a Lamp Data node linking to the same lamp object, including the Lamp Data node into a node group is recommended. Otherwise, when the mesh objects are imported to the other scene, all the materials may need to be modified.

Value Node

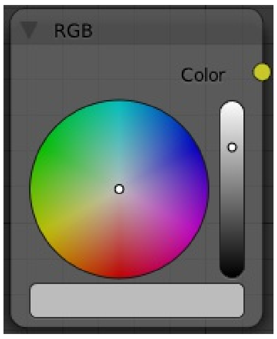


Value node

The Value node has no inputs; it just outputs a numerical value (floating point spanning 0.00 to 1.00) currently entered in the NumButton displayed in its controls selection.

Use this node to supply a constant, fixed value to other nodes' value or factor input sockets.

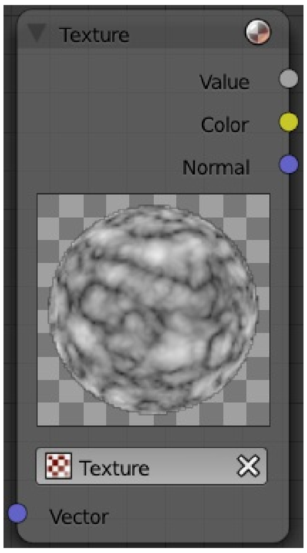
RGB Node



RGB node

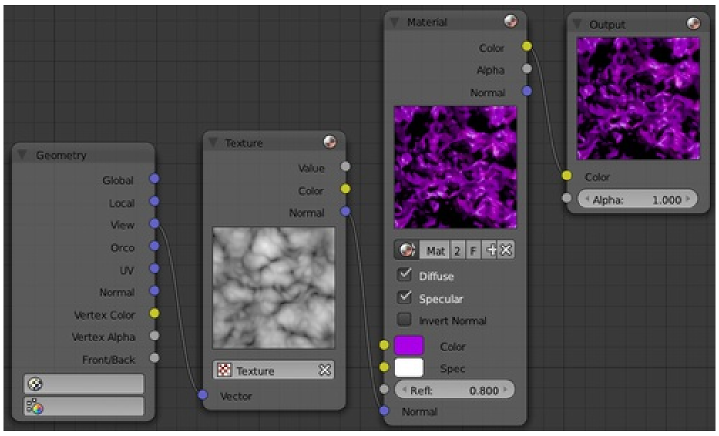
The RGB node has no inputs. It just outputs the value Color currently selected in its controls section.

Material Node "Texture"



Texture node

A texture, from the list of textures available in the current blend file, is selected and introduced through the value and/or color socket.



Example of the applying Texture node

Input

Vector
: Uses for map the texture to a specific geometric space.

Outputs

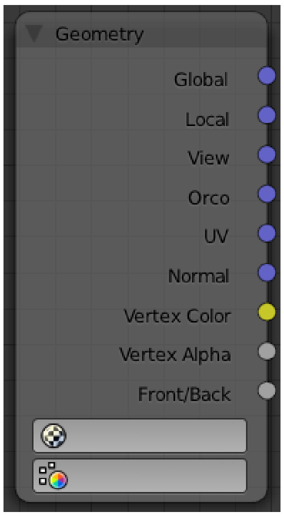
Value
: Straight black-and-white value of the texture, combined by the node.

Color
: Texture color output, combined by the node.

Normal
: Direction of normal texture, combined by the node.

In the example to the right, a cloud texture, as it would appear to a viewer, is added to a base purple material, giving a velvet effect. Note that you can have multiple texture input nodes. With nodes, you simply add the textures to the map and plug them into the map.

Geometry Node



Geometry node

The geometry node is used to specify how light reflects off the surface. This node is used to change a material's Normal response to lighting conditions.

Use this node to feed the Normal vector input on the Material node, to see how the material will look (i.e. shine, or reflect light) under different lighting conditions. Your choices are:

Global
: Global position of the surface.

Local
: Local position of the surface.

View
: Viewed position of the surface.

Orco
: Using the Original Coordinates of the mesh.

UV
: Using the UV coordinates of the mesh, selected in the field in bottom node.

Normal
: Surface Normal; On a flat plane with one light above and to the right reflecting off the surface.

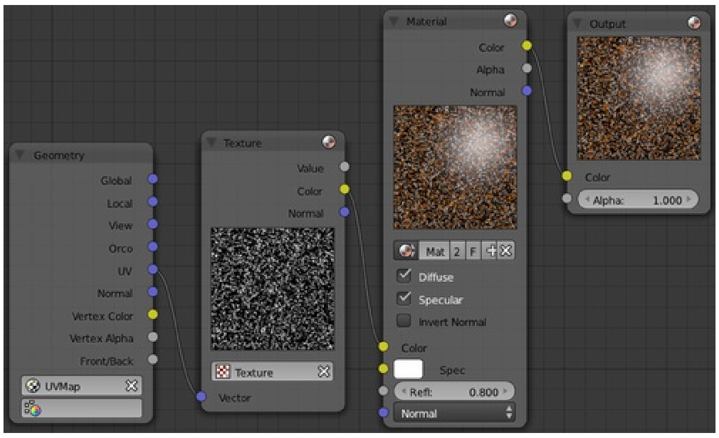
Vertex Color
: Allows for output value of group vertex colors, selected in the field in bottom node.

Vertex Alpha
: Allows for output alpha value of vertex.

Front/Back
: Allows for output to take into account front or back of surface is light relative the camera.

Note
These are exactly the same settings as in the Mapping panel for Textures, though a few settings - like Stress or Tangent - are missing here. Normally you would use this node as input for a Texture Node.

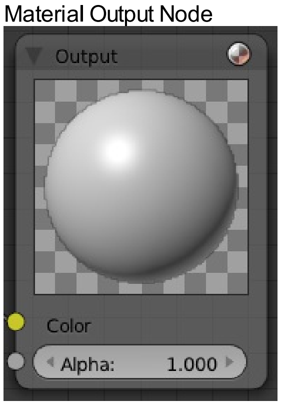
Geometry Node Example using a UV image



Setup to render an UV-Mapped Image Texture.

E.g.: To render an UV-mapped image, you would use the UV output and plug it into the Vector Input of a texture node. Then you plug the color output of the texture node into the color input of the material node - which corresponds to the setting on the Map To panel.

Material Output Node



Output material node

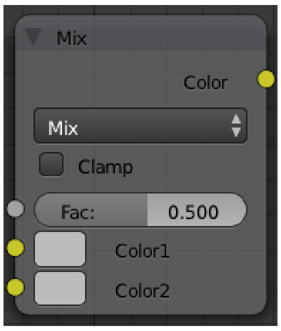
At any point, you may want to see the work in progress, especially right after some operation by a node. Simply create another thread from the output socket of the node to the picture input socket of an Output node to see a mini-picture.

Connect the alpha channel to set/see transparency.

Effective Output Node
The only Output node that is used for the Material in the end (i.e the only non-Preview) has a little **red sphere** on the upper right.

Material Color Nodes

MixRGB



MixRGB node

This node mixes a base color or image (threaded to the top socket) together with a second color or image (bottom socket) by working on the individual and corresponding pixels in the two images or surfaces. The way the output image is produced is selected in the drop-down menu. The size (output resolution) of the image produced by the mix node is the size of the base image. The alpha and Z channels (for compositing nodes) are mixed as well.

Not one, not two, but count 'em, sixteen mixing choices include:

Mix	The background pixel is covered by the foreground using alpha values.
Add	The pixels are added together. Fac controls how much of the second socket to add in. Gives a bright result. The "opposite" to Subtract mode.
Subtract	The foreground pixel (bottom socket) is subtracted from the background one. Gives a dark result. The "opposite" to Add mode.
Multiply	Returns a darker result than either pixel in most cases (except one of them equals white=1.0). Completely white layers do not change the background at all. Completely black layers give a black result. The "opposite" to Screen mode.
Screen	Both pixel values are inverted, multiplied by each other, the result is inverted again. This returns a brighter result than both input pixels in most cases (except one of them equals 0.0). Completely black layers do not change the background at all (and vice versa) - completely white layers give a white result. The "opposite" of Multiply mode.
Overlay	A combination of Screen and Multiply mode, depending on the base color.
Divide	The background pixel (top socket) is divided by the second one: if this one is white (= 1.0), the first one isn't changed; the darker the second one, the brighter is the result (division by 0.5 - median gray - is same as multiplication by 2.0); if the second is black (= 0.0, zero-division is impossible!), Blender doesn't modify the background pixel.
Difference	Both pixels are subtracted from one another, the absolute value is taken. So the result shows the distance between both parameters, black stands for equal colors, white for opposite colors (one is black, the other white). The result looks a bit strange in many cases. This mode can be used to invert parts of the base image, and to compare two images (results in black if they are equal).
Darken	Both pixels are compared to each other, the smaller one is taken. Completely white layers do not change the background at all, and completely black layers give a black result.
Lighten	Both parameters are compared to each other, the larger one is taken. Completely black layers do not change the image at all and white layers give a white result.
Dodge	Some kind of inverted Multiply mode (the multiplication is replaced by a division of the "inverse"). Results in lighter areas of the image.
Burn	Some kind of inverted Screen mode (the multiplication is replaced by a division of the "inverse"). Results in darker images, since the image is burned onto the paper, er..image (showing my age).
Color	Adds a color to a pixel, tinting the overall whole with the color. Use this to increase the tint of an image.
Value	The RGB values of both pixels are converted to HSV values. The values of both pixels are blended, and the hue and saturation of the base image is combined with the blended value and converted back to RGB.
Saturation	The RGB values of both pixels are converted to HSV values. The saturation of both pixels are blended, and the hue and value of the base image is combined with the blended saturation and converted back to RGB.
Hue	The RGB values of both pixels are converted to HSV values. The hue of both pixels are blended, and the value and saturation of the base image is combined with the blended hue and converted back to RGB.
Soft Light	Lightens or darkens base color depending on the blend color brightness. The effect is softer than that of Linear Light or Overlay modes, with pure white and pure black blend colors not yielding pure white/black results.
Linear Light	Brightens base color depending on blend color. If blend color is more than 50% bright, base color is brightened by the blend color values, otherwise it is darkened by the blend color values.

Inputs

Fac
: The amount of mixing of the bottom socket is selected by the Factor input field (Fac:). A factor of zero does not use the bottom socket, whereas a value of 1.0 makes full use. In Mix mode, 50:50 (0.50) is an even mix between the two, but in Add mode, 0.50 means that only half of the second socket's influence will be applied.

Color 1
: Input color value. The value can be provided by another node or set manually. Includes a color swatch, allowing you to select the color directly on the node.

Color 2

Input color value. The value can be provided by another node or set manually. Includes a color swatch, allowing you to select the color directly on the node.

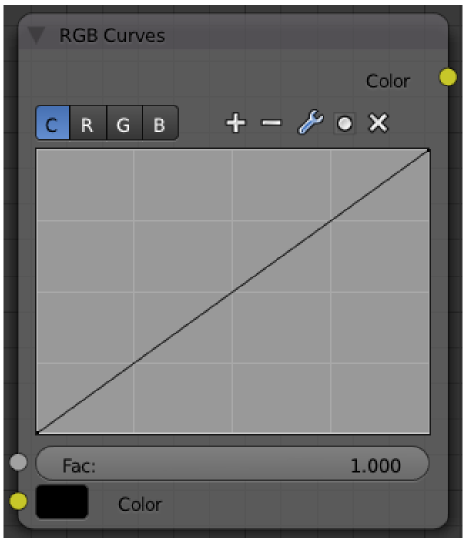
Outputs

Color
: Value of the color, combined by the node.

Controls

Clamp
: Clamp result of the node to 0...1 range.

RGB Curves



RGB Curves node

For each color component channel (RGB) or the composite (C), this node allows you to define a bezier curve that varies the input (across the bottom, or x-axis) to produce an output value (the y-axis). By default, it is a straight line with a constant slope, so that .5 along the x-axis results in a .5 y-axis output. Click and drag along the curve to create a control point and to change the curve's shape. Use the X to delete the selected (white) point.

Clicking on each C R G B component displays the curve for that channel. For example, making the composite curve flatter (by clicking and dragging the left-hand point of the curve up) means that a little amount of color will result in a lot more color (a higher Y value). Effectively, this bolsters the faint details while reducing overall contrast. You can also set a curve just for the red, and for example, set the curve so that a little red does not show at all, but a lot of red does.

Inputs

Fac
: Factor. The degree of node's influence in node tree. The value can be provided by another node or set manually. Value range - from «-1» (inverted effect) to «1».

Color
: Input color value. The value can be provided by another node or set manually. Includes a color swatch, allowing you to select the color directly on the node.

Outputs

Color
: Value of the color, combined by the node.

Controls

Curve channel selector

Channel selector
: Allows to select appropriate curve channel.

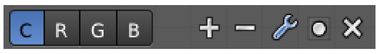
C

Composite curve.

R
: Red channel curve.

G
: Green channel curve.

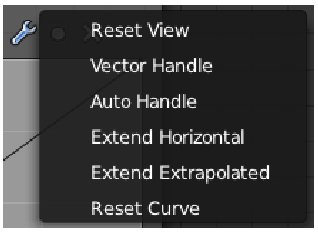
B
: Blue channel curve.



Node curve controls

+
: Zoom in curve.

−
: Zoom out curve.



Advanced tools for curve

Reset View
: Resets view of the cuve.

Vector Handle
: Vector type of curve point's handle.

Auto Handle
: Automatic type of curve point's handle.

Extend Horizontal
: Extends the curve horizontal.

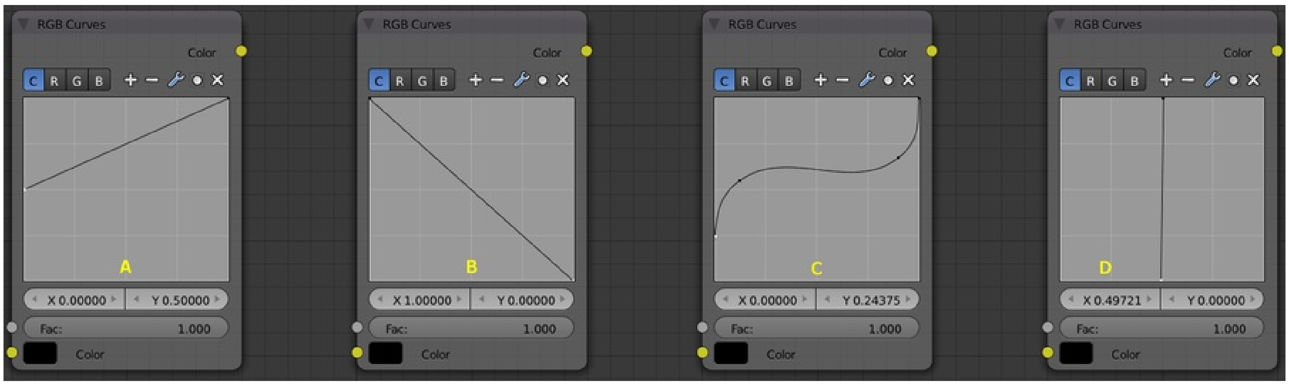
Extend Extrapolated
: Extends the curve extrapolated.

Reset Curve
: Resets the curve in default (removes all added curve's points).

●
: Clipping options display of the curve.

✕
: Deletes points of the curve.

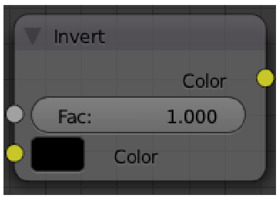
Here are some common curves you can use to achieve desired effects:



A: X 0.00000, Y 0.50000, Fac: 1.000
B: X 1.00000, Y 0.00000, Fac: 1.000
C: X 0.00000, Y 0.24375, Fac: 1.000
D: X 0.49721, Y 0.00000, Fac: 1.000

A) Lighten **B**) Negative **C**) Decrease Contrast **D**) Posterize

Invert



Invert node

This node simply inverts the input values and colors.

Inputs

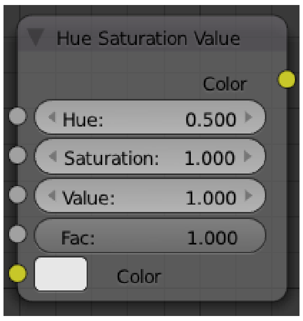
Fac
: Factor. The degree of node's influence in node tree. The value can be provided by another node or set manually.

Color
: Input color value. The value can be provided by another node or set manually. Includes a color swatch, allowing you to select the color directly on the node.

Outputs

Color
: Value of the color, combined by the node.

Hue Saturation Value



Hue Saturation Value node

Use this node to adjust the Hue, Saturation, and Value of an input.

Inputs

Fac
: Factor. The degree of node's influence in node tree. The value can be provided by another node or set manually.

Hue
: Input hue value of color. The value can be provided by another node or set manually.

Saturation
: Input saturation value of color . The value can be provided by another node or set manually.

Value
: Input HSV-Value of color. The value can be provided by another node or set manually.

Fac
: Factor. The degree of node's influence in node tree. The value can be provided by another node or set manually.

Color
: Input color value. The value can be provided by another node or set manually. Includes a color swatch, allowing you to select the color directly on the node.

Outputs

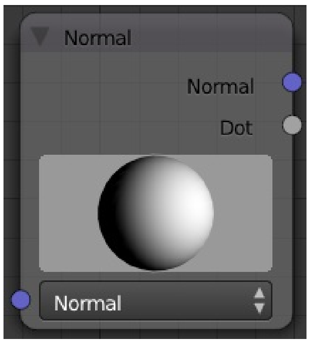
Color
: Value of the color, combined by the node.

Material Vector Nodes

Vector nodes manipulate information about how light interacts with the material, multiplying vector sets, and other wonderful things that normal humans barely comprehend (except math geniuses, who may not be considered «normal»). Even if you aren't a math wiz, you'll find these nodes to be very useful.

Vectors, in general, are two or three element values, for example, surface normals are vectors. Vectors are also important for calculating shading.

Normal Node



Normal node

The Normal node generates a normal vector and a dot product. Click and Drag on the sphere to set the direction of the normal.

This node can be used to input a new normal vector into the mix. For example, use this node as an input to a Color Mix node. Use an Image input as the other input to the Mixer. The resulting colorized output can be easily varied by moving the light source (click and dragging the sphere).

The (face) normal is the direction of the face in relation to the camera. You can use it to do the following:

- Use this node to create a fixed direction -> output Normal.
- Calcuate the Dot-Product with the Normal-Input. The Dot-Product is a scalar value (a number).
  - If two normals are pointing in the same direction the Dot-Product is 1.
  - If they are perpendicular the Dot-Product is zero (0).
  - If they are antiparallel (facing directly away from each other) the Dot-Product is -1. *And you never thought you would use the Vector Calculus class you took in college - shame on you!*

So now we can do all sorts of things that depends on the viewing angle (like electron scanning microscope effect). And the best thing about it is that you can manipulate the direction interactively.

One caveat
The normal is evaluated per face, not per pixel. So you need enough faces, or else you don't get a smooth result

Inputs

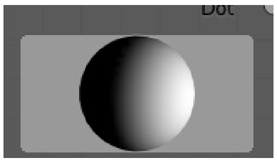
Normal
: 3D-direction of the face in relation to the camera. The value can be provided by another node or set manually.

Outputs

Normal
: Fixed 3D-direction, combined by the node.

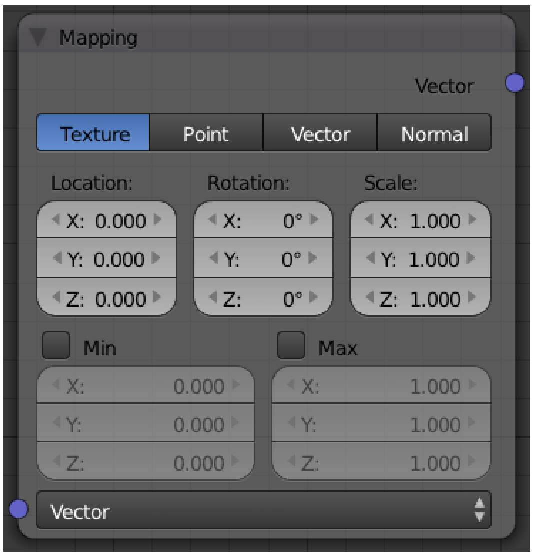
Dot
: Scalar value (a number), combined by the node.

Controls


Interactive Normal node preview

Interactive node preview
: Allows click and drag on the sphere in node center to set the direction of the normal.

Mapping Node



Mapping node

Essentially mapping node allows the user to modify a mapping of system of 3D-coordinates. Typically used for modifying texture coordinates.

Mapping can be rotated and clamped if desired.

Inputs

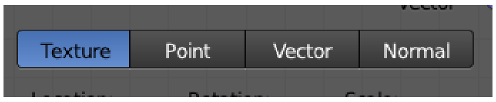
Vector
: The input vector (3D-direction in relation to the camera) of some the coordinates' mapping. The value can be provided by another node or set manually.

Outputs

Vector
: The output vector, combined by the node.

Controls

The controls of the node have been ordered in X, Y, Z order. If you want to use the clamping options, try enabling Min and Max.



Mapping Node Vector Types controls

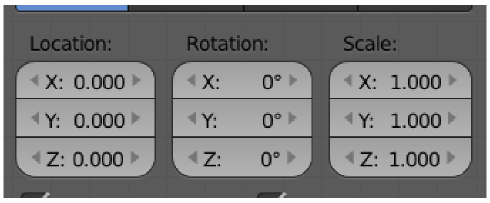
Vector type selector
: Type of vector that the mapping transforms.

  Texture
  : Transform a texture by inverse mapping the texture coordinates.

  Point
  : Transform a point.

  Vector
  : Transform a direction vector.

  Normal
  : Transform a normal vector with unit length.



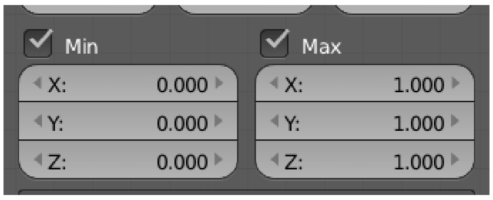
Mapping Node Transforms controls

Location
: Transform position vector.

Rotation
: Transform rotation vector.

Scale

Transform scale vector.

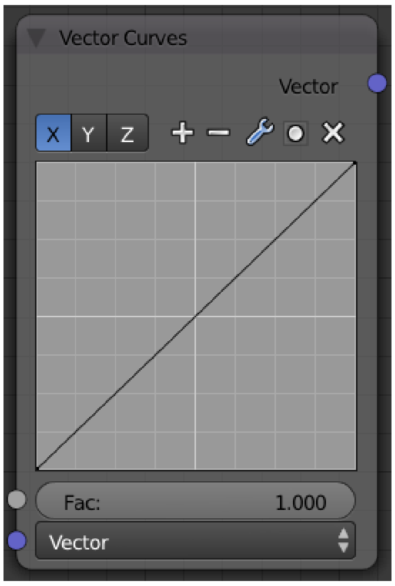


Mapping Node Clipping controls

Min
: Minimum clipping value.

Max
: Maximum clipping value.

Vector Curves



Vector Curves node

The Vector Curves node maps an input vector x, y, and z components to a diagonal curve. Use this node to remap a vector value using curve controls.

Click and drag along the curve to create a control point and to change the curve's shape. Use the X to delete the selected (white) point.

Inputs

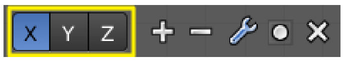
Fac
: Factor. The degree of node's influence in node tree. The value can be provided by another node or set manually.

Vector
: The input vector (3D-direction in relation to the camera). The value can be provided by another node or set manually.

Outputs

Vector
: The output vector, combined by the node.

Controls



Curve channel selector

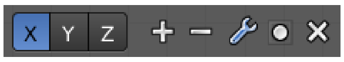
Channel selector
: Allows to select appropriate curve channel.

X
: Curve of X-direction.

Y

Curve of Y-direction.

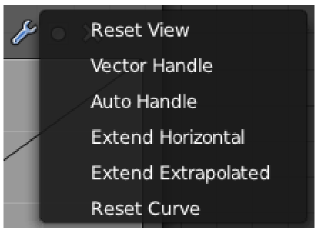
Z
: Curve of Z-direction.



Node curve controls

+
: Zoom in curve.

−
: Zoom out curve.



Advanced tools for curve

Reset View
: Resets view of the cuve.

Vector Handle
: Vector type of curve point's handle.

Auto Handle
: Automatic type of curve point's handle.

Extend Horizontal
: Extends the curve horizontal.

Extend Extrapolated
: Extends the curve extrapolated.

Reset Curve
: Resets the curve in default (removes all added curve's points).

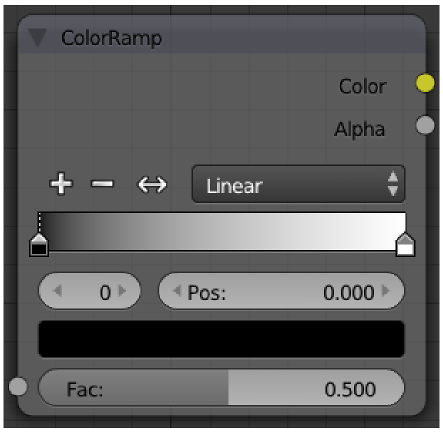
●
: Clipping options display of the curve.

✕
: Deletes points of the curve.

Material Convertor Nodes

As the name implies, these nodes convert the colors in the material in some way.

ColorRamp Node



ColorRamp node

The ColorRamp Node is used for mapping values to colors with the use of a gradient. It works exactly the same way as a Colorband for textures and materials, using the Factor value as a slider or index to the color ramp shown, and outputting a color value and an alpha value from the output sockets.

By default, the ColorRamp is added to the node map with two colors at opposite ends of the spectrum. A completely black black is on the left (Black as shown in the swatch with an Alpha value of 1.00) and a whitewash white is on the right.

To select a color, LMB click on the thin vertical line/band within the colorband. The example picture shows the black color selected, as it is highlighted white. The settings for the color are shown above the colorband as (left to right): color swatch, Alpha setting, and interpolation type.

Inputs

Fac
: Factor. The degree of node's influence in node tree. The value can be provided by another node or set manually.

Outputs

Color
: Value of the color, combined by the node.

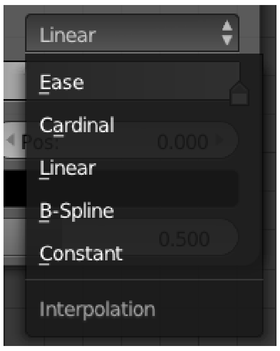
Alpha
: Value of the alpha, combined by the node.

Controls

+
: Add a new mark to the center of the colorband with the default color (neutral gray).

−
: Remove the currently selected mark from the colorband.

↔
: Flip the colorband.



Modes of interpolation between marker's values color ramp

Interpolation

Various modes of interpolation between marker's values can be chosen in the Interpolation menu:

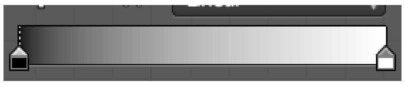
Ease
: Ease by quadratic equation.

Cardinal
: Cardinal.

Linear
: Linear (default). A smooth, consistent transition between colors.

B-Spline
: B-Spline.

Constant
: Constant.


Colorband

Colorband
: Contain a gradient through a sequence of many colors (with alpha), each color acting across a certain position in the spectrum.

The number of the active mark.

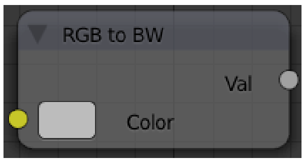
Pos. The position of the active color mark in the colorband (range 0.0–1.0). The position of the color marks can also be changed by LMB dragging them in the colorband.

Color swatch to color selection for a mark

Color Selector
: Allows set color and alpha values for each marker.

See more details about node controls' functions here.

RGB to BW Node



RGB to BW node

This node converts a color image to black-and-white.

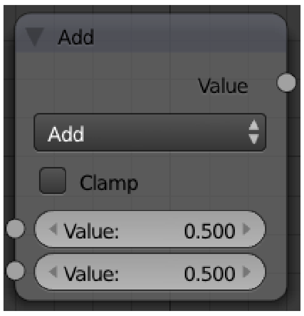
Inputs

Color
: Input color value. Includes a color swatch, allowing you to select the color directly on the node.

Outputs

Value
: Black-and-white value of the input color, converted by the node.

Math Node



Math node

This node performs the selected math operation on an image or buffer. All common math functions are supported. If only an image is fed to one Value socket, the math function will apply the other Value consistently to every pixel in producing the output Value. Select the math function by clicking the up-down selector where the "Add" selection is shown.

Inputs

Value
: Input value 1 (upper). The value can be provided by another node or set manually.

Value
: Input value 2 (lower). The value can be provided by another node or set manually.

Outputs

Value
: Output value, converted by the node.

Controls

Clamp
: Clamps the result between 0 and 1.

Operation
: Selector the math function for conversion.

  Add
  : Add the two inputs

  Subtract
  : Subtract input 2 from input 1

  Multiply
  : Multiply the two inputs

  Divide
  : Divide input 1 by input 2

  Sine
  : The sine of input 1 (degrees)

  Cosine
  : The cosine of input 1 (degrees)

  Tangent
  : The tangent of input 1 (degrees)

  Arcsine
  : The arcsine (inverse sine) of input 1 (degrees)

  Arccosine
  : The arccosine (inverse cosine) of input 1 (degrees)

  Arctangent
  : The arctangent (inverse tangent) of input 1 (degrees)

  Power
  : Input 1 to the power of input 2 (input1^input2)

  Logarithm
  : Log base input 2 of input 1

  Minimum
  : The minimum of input 1 and input 2

  Maximum
  : The maximum of input 1 and input 2

  Round
  : Rounds input 1 to the nearest integer

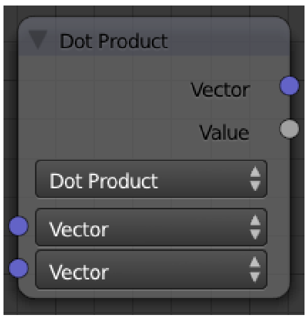
  Less Than
  : Test if input 1 is less than input 2, returns 1 for true, 0 for false

  Greater Than
  : Test if input 1 is greater than input 2, returns 1 for true, 0 for false

  Modulo
  : Division of input 1 by input 2 with remainder.

  Absolute
  : Always return non-negative value from any operation input 2 between input 1.

Vector Math Node



Vector Math node

This node performs the selected math operation on vectors. Select the math function by clicking the up-down selector where the "Add" selection is shown.

Inputs

Vector
: Input vector 1 (upper). The value can be provided by another node or set manually.

Vector
: Input vector 2 (lower). The value can be provided by another node or set manually.

Outputs

Vector
: Output vector, converted by the node.

Value
: Output value, converted by the node.

Controls

Operation
: Selector the math function for conversion.

  Add
  : Adding input 1 and 2.

  Subtract
  : Subtracting input 1 and 2.

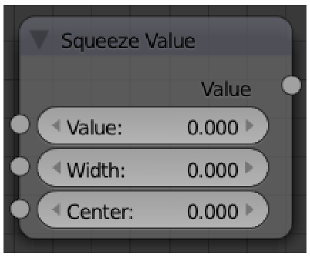
  Average
  : Averaging input 1 and 2.

  Dot Product
  : Algebraic operation that takes two equal-length sequences of vectors 1 and 2 and returns a single number. Result – scalar.

  Cross Product
  : Geometric binary operation on two vectors 1 and 2 in three-dimensional space. It results in a vector which is perpendicular to both and therefore normal to the plane containing them. Result – vector.

  Normalize
  : Normalizing input 1 and 2.

Squeeze Value Node



Value: 0.000
Width: 0.000
Center: 0.000

Squeeze Value node

This node is used primarily in conjunction with the Camera Data node used. The camera data generate large output values, both in terms of the depth information as well as the extent in the width. With the squeeze Node high output values to an acceptable material for the node degree, ie to values between 0.0 - 1.0 scaled down.

Inputs

Value
: Any numeric value. The value can be provided by another node or set manually.

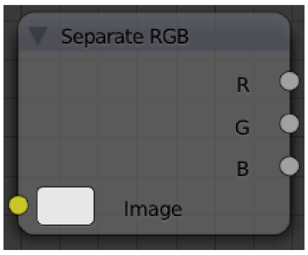
Width
: Determines the curve between sharp S-shaped (width = 1) and stretched (Width = 0.1). Negative values reverse the course. The value can be provided by another node or set manually.

Center
: The center of the output value range. This input value is replaced by the output value of 0.5. The value can be provided by another node or set manually.

Outputs

Value
: A value between 0 and 1, converted by the node.

Separate RGB Node



Separate RGB node

This node separates an image into its red, green, blue channels - traditional primary colors, also broadcast directly to most computer monitors.

Inputs

Image
: Input color value. Includes a color swatch, allowing you to select the color directly on the node.

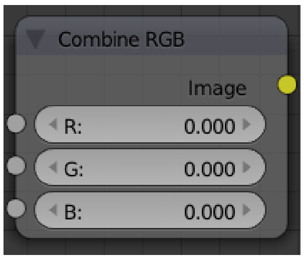
Outputs

R
: Value of the red color channel, separated out by the node.

G
: Value of the green color channel, separated out by the node.

B
: Value of the blue color channel, separated out by the node.

Combine RGB Node



Combine RGB node

This node combines a color (image) from separated red, green, blue channels.

Inputs

R
: Input value of red color channel. The value can be provided by another node or set manually.

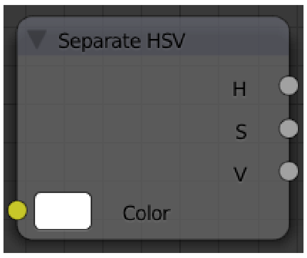
G
: Input value of green color channel. The value can be provided by another node or set manually.

B
: Input value of blue color channel. The value can be provided by another node or set manually.

Outputs

Image
: Output value of the color, combined by the node.

Separate HSV Node



Separate HSV node

This node separates an image into image maps for the hue, saturation, value channels. Three values, often considered as more intuitive than the RGB system (nearly only used on computers)

Use and manipulate the separated channels for different purposes; i.e. to achieve some compositing/color adjustment result. For example, you could expand the Value channel (by using the multiply node) to make all the colors brighter. You could make an image more relaxed by diminishing (via the divide or map value node) the Saturation channel. You could isolate a specific range of colors (by clipping the Hue channel via the Colorramp node) and change their color (by the Add/Subtract mix node).

Inputs

Color
: Input color value. Includes a color swatch, allowing you to select the color directly on the node.

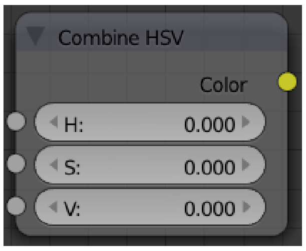
Outputs

H
: Value of the hue color channel, separated out by the node (in some way, choose a «color» of the rainbow).

S
: Value of the saturation color channel, separated out by the node (the *quantity* of hue in the color (from desaturate - shade of gray - to saturate - brighter colors)).

V
: Value of the value color channel, separated out by the node (the **luminosity** of the color (from 'no light' - black - to 'full light' - 'full' color, or white if Saturation is 0.0)).

Combine HSV Node



Combine HSV node

This node combines a color from separated hue, saturation, value color channels.

Inputs

H
: Input value of hue color channel. The value can be provided by another node or set manually.

S
: Input value of saturation color channel. The value can be provided by another node or set manually.

V
: Input value of value color channel. The value can be provided by another node or set manually.

Outputs

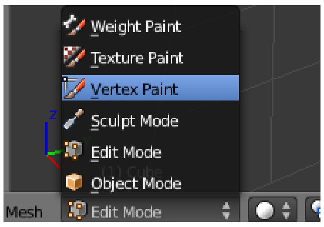
Color
: Output value of the color, combined by the node.

Vertex Painting

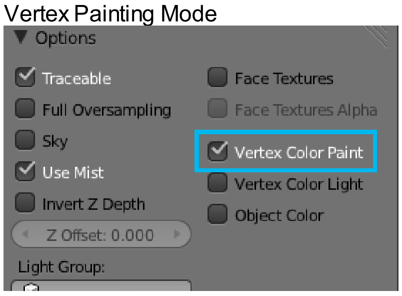
Vertex Painting is a simple way of painting color onto an object, by directly manipulating the color of vertices, rather than textures, and is fairly straightforward.

When a vertex is painted, the colour of the vertex is modified according to the rules of the 'brush'. The color of all visible planes and edges attached to the vertex are then modified with a gradient to the color of the other connected vertices. (Note that the color of non-visible faces are not modified).

Vertex colors can be painted by first going into Edit Mode, then switching to Vertex Paint Mode; however, it will not show up in the render unless you check "Vertex Color Paint" in the Materials Options Panel.



Vertex Painting Mode

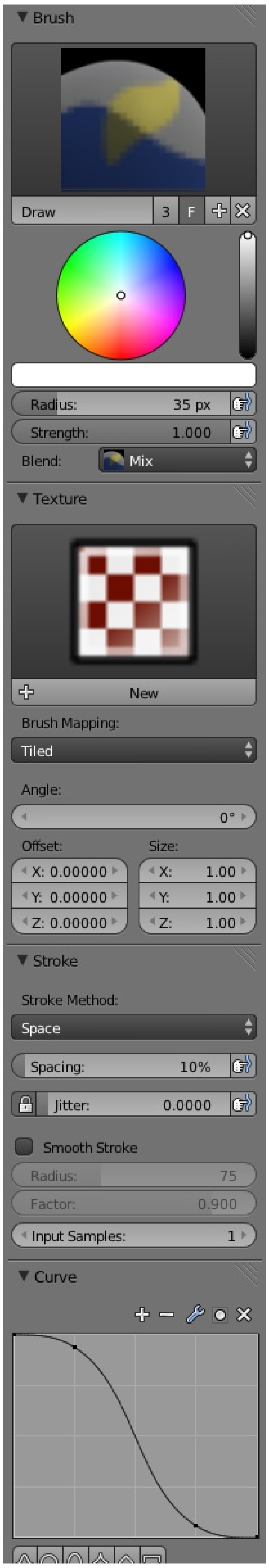

Options

- [✓] Traceable
- [ ] Full Oversampling
- [ ] Sky
- [✓] Use Mist
- [ ] Invert Z Depth
- Z Offset: 0.000
- Light Group:
- [ ] Face Textures
- [ ] Face Textures Alpha
- [✓] Vertex Color Paint
- [ ] Vertex Color Light
- [ ] Object Color

Check this box

Settings

The Tools Shelf, shortcut T contains most of the options for vertex painting. The following sections describe the controls in each of the available panels.



Brush

Draw 3 F

Radius: 35 px

Strength: 1.000

Blend: Mix

Texture

New

Brush Mapping:

Tiled

Angle:

0°

Offset:

X: 0.00000

Y: 0.00000

Z: 0.00000

Size:

X: 1.00

Y: 1.00

Z: 1.00

Stroke

Stroke Method:

Space

Spacing: 10%

Jitter: 0.0000

Smooth Stroke

Radius: 75

Factor: 0.900

Input Samples: 1

Curve

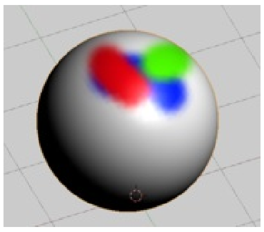
Settings for vertex painting

Brush

Brush Datablock
: The image, name panel and color selector at the top allows you to select brush presets, rename brushes, as well as add custom brushes, and delete them.

Radius
: Set the radius of the brush

Strength
: Set the strength of the brush's effect.


Mix overlay with full strength

Blend menu

Mix
: Mixes RGB values. When set to a strength of 1.0, it will cover the underlying "paint".

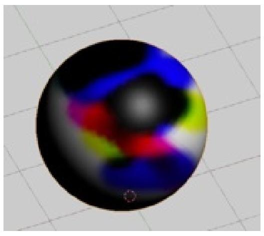
Add
: Adds RGB values. Will eventually turn the entire object white as RGB values accumulate to 1.0-1.0-1.0: Pure White.

Subtract
: Subtracts RGB values. Usually results in Black.

Multiply
: Multiplies brush colors by the vertex colors.

Blur
: Blurs vertex colors.

Lighten
: Lightens the color of the vertices.


Subtract with full strength

Darken
: Darkens the color of the vertices.

Texture

Use the texture selector at the bottom of the paint panel to select a pre-loaded image or procedural texture to use as your brush pattern. Note that in order to use it, you must have a placeholder material defined, and that particular texture defined using the Material and Texture buttons. It is not necessary to have that material or texture applied to any mesh anywhere; it must only be defined.

Brush Mapping Mode
: Sets how the texture is applied to the brush

  View Plane

  : In 2D painting, the texture moves with the brush

  Tiled

  : The texture is offset by the brush location

  3D

  : Same as tiled mode

  Stencil

  : Texture is applied only in borders of the stencil.

  Random

  : Random applying of texture.

Angle
: This is the rotation angle of the texture brush. It can be changed interactively via CtrlF in the 3D view. While in the interactive rotation you can enter a value numerically as well. Can be set to:

  User
  : Directly input the angle value.

  Rake
  : Angle follows the direction of the brush stroke. Not available with 3D textures.

  Random
  : Angle is randomized.

Offset
: Offset the texture in x, y, and z.

Size
: Set the scale of the texture in each axis.

Stroke

Stroke Method
: Allows set the way applying strokes.

  Airbrush
  : Flow of the brush continues as long as the mouse click is held, determined by the Rate setting. If disabled, the brush only modifies the color when the brush changes its location.

    Rate
    : Interval between paints for airbrush

  Space
  : Creates brush stroke as a series of dots, whose spacing is determined by the Spacing setting.

    Spacing
    : Represents the percentage of the brush diameter. Limit brush application to the distance specified by spacing.

  Dots
  : Apply paint on each mouse move step

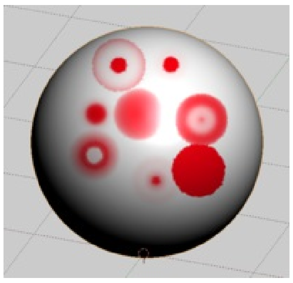
  Jitter
  : Jitter the position of the brush while painting

Smooth stroke
: Brush lags behind mouse and follows a smoother path. When enabled, the following become active:

  Radius
  : Sets the minimun distance from the last point before stroke continues.

  Factor
  : Sets the amount of smoothing.

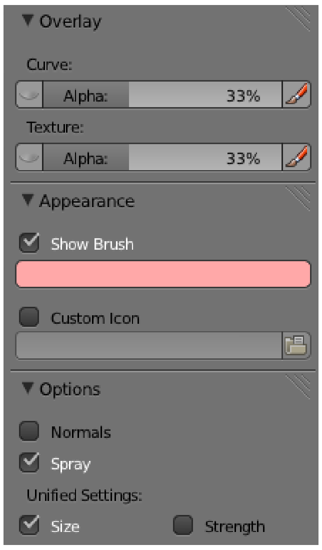
Input Samples
: Average multiple input samples together to smooth the brush stroke.


Various brush curves

Curve

Brush Curves affect how strongly the color is applied depending on distance from the center of the brush. In other words, they allow you to edit the Falloff of the brush intensity.

Options


Overlay
Curve:
Alpha: 33%
Texture:
Alpha: 33%
Appearance
✓ Show Brush
Custom Icon
Options
Normals
✓ Spray
Unified Settings:
✓ Size
Strength

Options for vertex painting

Overlay

Allows you to customize the display of curve and texture that applied to the brush.

Appearance

Allows you to customize the color of the brush radius outline, as well as specify a custom icon.

Options

Normals
: Applies the Vertex Normal before painting. This does not usually affect painting.

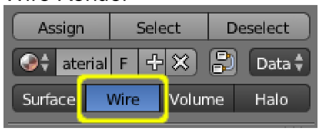
Spray
: Continues painting for as long as the mouse is held.

Unified Settings

Size
: All brushes use the same size.

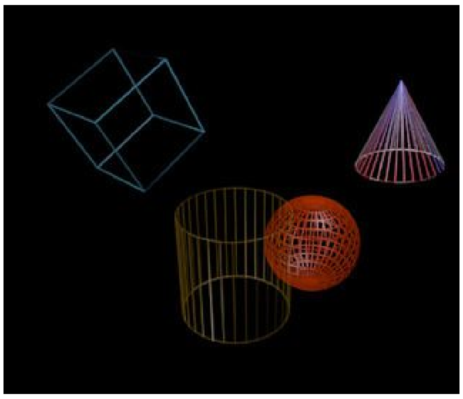
Strength
: All brushes use the same strength.

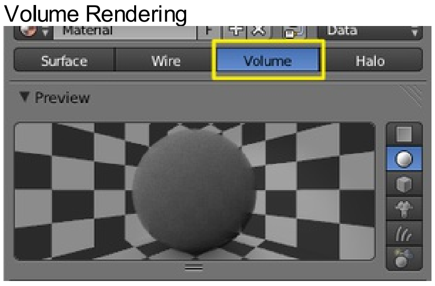
Wire Render



Wire Render

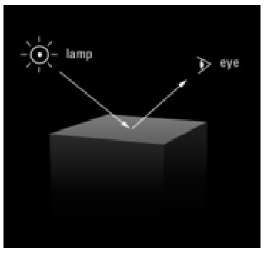
The Wire Render option in the Materials section provides a way of showing a rendered image of the edges in an object. Each edge is rendered as a single-pixel image of the edges which make up the mesh. The colors, alpha and other relevant properties of the lines are selected with the same control panels as provided by the Surface rendered image.

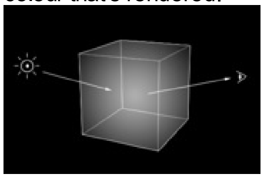

Wire Render

Volume Rendering


Activation volume rendering

Volume rendering is a method for rendering light as it passes through participating media, within a 3d region. The implementation in Blender's sim-physics branch is a physically based model, which represents the various interactions of light in a volume relatively realistically.

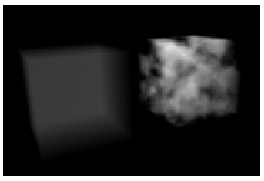


Solid rendering
The process of rendering a solid surface involves the camera finding a piece of geometry, then calculating the light that bounces from light sources (lamp objects, or other geometry), off the surface, and towards the camera. The light that arrives at the camera is the final colour that's rendered.

Volume rendering
Rendering a volume works differently. Light enters a 3D region of space (defined as the volume) that may be filled with small particles, such as smoke, mist or clouds.

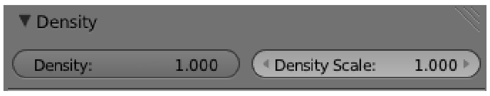
The light bounces around off the various molecules, being scattered or absorbed, until some light passes through the volume and reaches the camera. In order for that volume to be visible, the renderer must figure out how much material the light has passed through and how it has acted and reacted within that volume, the volume object needs to contain a 3D region of space, for example a watertight closed mesh, such as a cube, not just a flat surface like a plane. To get an image, the renderer has to step through that region, and see how much 'stuff' is there (density) in order to see how light is absorbed or scattered or whatever. This can be a time consuming process since it has to check a lot of points in space and evaluate the density at each.

Options

Density


Constant density vs
textured density

Many things can happen to the light as it passes through the volume, which will influence the final colour that arrives at the camera. These represent physical interactions that happen in the real world, and most of these are dependent on the density of the volume, which can either be a constant density throughout, or varied, controlled by a texture. It is by controlling the density that one can get the typical 'volumetric' effects such as clouds or thick smoke.


Density
Density: 1.000
Density Scale: 1.000

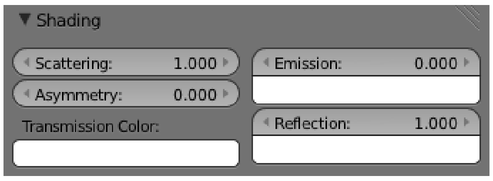
Density options

Density
The base density of the material - other density from textures is added on top
Density Scale
A global multiplier to increase or decrease the apparent density. This can be useful for getting consistent results across different scene scales.

Shading

Spot lamp scattering in a constant volume

When light enters a volume from an external source, it doesn't just pass straight through. Light gets scattered off tiny particles in the volume, and some proportion of that light reaches the camera. This property makes it possible to see light beams as they travel though a volume and are scattered towards the eye.

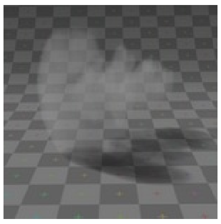

▼ Shading

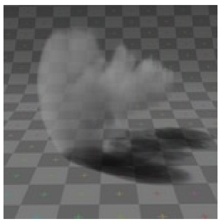
Scattering: 1.000 | Emission: 0.000
Asymmetry: 0.000
Transmission Color: | Reflection: 1.000

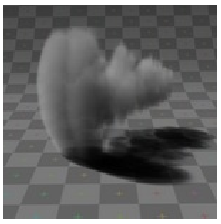
Shading options

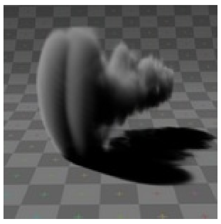
Scattering
: The amount of light that is scattered out of the volume. The more light that is scattered out of the volume, the less it will penetrate through the rest of the volume. Raising this parameter can have the effect of making the volume seem denser, as the light is scattered out quickly at the 'surface' of the volume, leaving the areas internal to the volume darker, as the light doesn't reach it.

Note in the examples below, the less light that is scattered out of the volume, the more easily it penetrates throughout the volume and to the shadow.

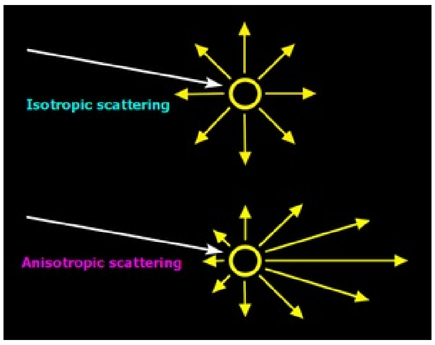

Scattering: 0.5


Scattering: 1.0


Scattering: 2.0


Scattering: 5.0

Asymmetry



Isotropic and Anisotropic scattering

The default method for scattering light in a volume is for the light to be deflected evenly in all directions - known as Isotropic scattering. In real life different types of media can scatter light in different angular directions, known as Anisotropic scattering. Back-scattering means that light is scattered more towards the incoming light direction, and forward-scattering means it's scattered along the same direction as the light is travelling.

Asymmetry
: Asymmetry controls the range between back-scattering (-1.0) and forward-scattering (1.0). The default value of 0.0 gives Isotropic scattering (even in all directions).

Transmission

Transmission is a general term for light that is transmitted throughout a volume.

This transmitted light can be the result of various different interactions, for example:

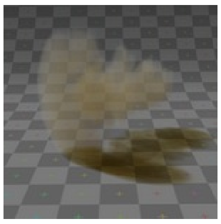
- the left over result of incoming light after it has reflected/scattered out of the volume

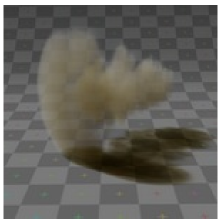
- the left over result of light after being absorbed by the volume (and converted to heat)

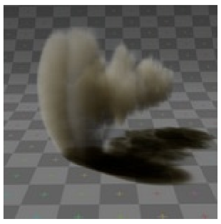
Here, the transmission colour is used to set the end result colour that light becomes after it is transmitted through the volume.

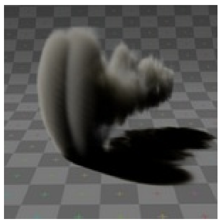
Transmission Color
: The resultant colour of light that is transmitted through the volume.

Note in the examples below, as more light is scattered out of the volume, there is less available to be transmitted through.


Transmission color: Yellow, Scattering: 0.5


Transmission color: Yellow, Scattering: 1.0


Transmission color: Yellow, Scattering: 2.0


Transmission color: Yellow, Scattering: 5.0

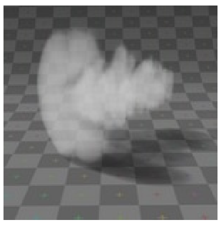
Emission

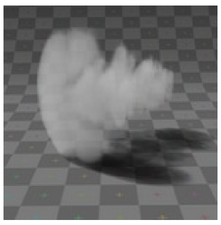
Some volumes can emit light where there was none before, via chemical or thermal processes, such as fire. This light is generated from the volume itself and is independent of light coming from external sources.

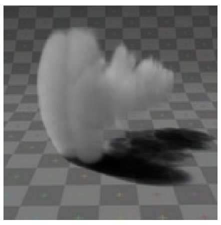
Currently, this emitted light does not affect other volumes or surfaces (similar to surface material type, 'Emit' option).

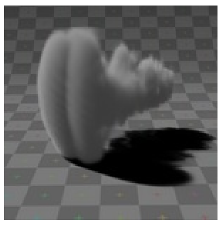
Emission Color
: The colour of light that is emitted by the volume.

Emission
: An intensity multiplier for the emitted colour, for scaling up and down.


Emission 0.25, Scattering: 0.5


Emission 0.25, Scattering: 1.0


Emission 0.25, Scattering: 2.0


Emission 0.25, Scattering: 5.0

Reflection

The 'reflection' parameters can be used to tint or scale the light that's scattered out of the volume. This only affects light that has come from lamps and been scattered out, it doesn't affect the colour of transmitted or emitted light and is .

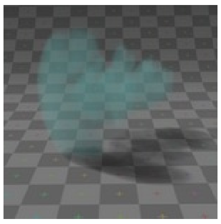
These settings are not physically correct because they don't conserve energy - the light scattering out doesn't affect the remaining light that is transmitted throughout the rest of the volume. For example, physically speaking, if the orange components of the light are scattered out of the volume towards the camera, only the inverse of that (blue) will remain to continue penetrating through the volume, causing the volume to take on a multi-coloured appearance, which can be difficult to use. To make it a bit easier to plainly set the colour of the volume, you can use the reflection parameters to quickly set an overall tint.

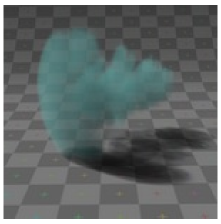
Reflection Color
: The colour of light that is scattered out of the volume.

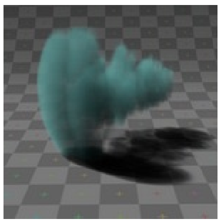
Reflection
: An intensity multiplier for the reflection, for scaling up and down.

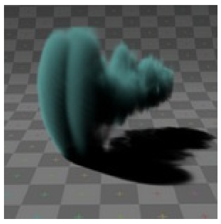
Hints

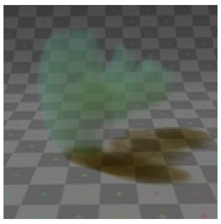
Ideally try to accomplish as much as you can with the other volume settings and lighting before using the reflection controls. If you stick to what's physically plausible, the material will act correctly, and be more predictable and usable in a wider range of lighting scenarios. Of course you can always break the rules too!


Reflection: Green, Scattering: 0.5

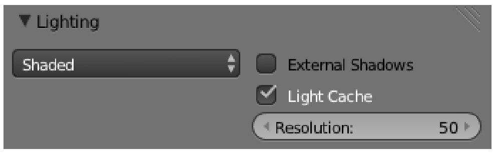

Reflection: Green, Scattering: 1.0


Reflection: Green, Scattering: 2.0


Reflection: Green, Scattering: 5.0

			
Reflection: Green, Transmission: Yellow, Scattering: 0.5	Reflection: Green, Transmission: Yellow, Scattering: 1.0	Reflection: Green, Transmission: Yellow, Scattering: 2.0	Reflection: Green, Transmission: Yellow, Scattering: 5.0

Lighting



Lighting options

Several shading modes are available, providing a range of options between fast to render and physically accurate.

Lighting Mode

Shadeless
: Shadeless is the simplest, useful for thin, wispy mist or steam.

Shadowed
: Shadowed is similar, but with shadows of external objects.

Shaded

: Shaded uses a volumetric single-scattering method, for self-shading the volume as light penetrates through.

Multiple Scattering
: Allows multiple scatter calculations.

Shaded+Multiple Scattering
: Combines Shaded and Multiple Scattering functionality.

Shaded Options:

External Shadows
: Receive shadows from sources outside the volume (temporary).

Light Cache
: Pre-calculate the shading information into a voxel grid, speeds up shading at slightly less accuracy.

Resolution
: Resolution of the voxel grid, low resolutions are faster, high resolutions use more memory.

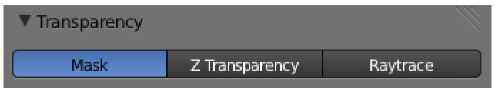
Multiple Scattering Options:

Diffusion
: Diffusion factor, the strength of the blurring effect.

Spread
: Proportional distance over which the light is diffused.

Intensity
: Multiplier for multiple scattered light energy.

Transparency



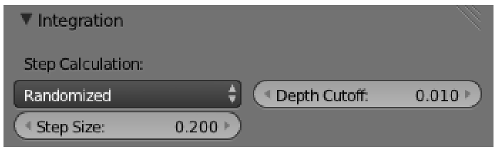
Transparency options

Mask
: Mask the Background.

Z Transparency
: Use Alpha buffer for transparent faces.

Raytrace
: Use Raytracing for Transparent Refraction rendering.

Integration



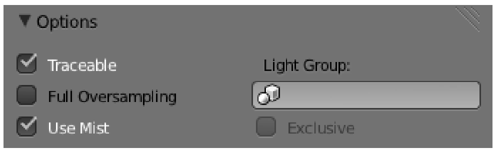
Integration options

Step Calculation Method
Method of calculating the step through the volume.

Randomized
Randomized method of calculating the step.
Constant
Constant method of calculating the step.

Step Size
Distance between subsequent volume depth samples. Step Sizes determine how noisy the volume is. Higher values result in lower render times and higher noise.
Depth Cutoff
Stop ray marching early if transmission drops below this luminance - higher values give speedups in dense volumes at the expense of accuracy.

Options



Material volume options

Traceable
Allow this material to calculate raytracing.
Full Oversample
Force this material to render full shading/textures for all anti-aliasing samples.
Use Mist
Use mist with this material (in world settings).

Light Group
Limit lighting of this material to lamps in this group.
Exclusive
Material uses this group exclusively. Lamps are excluded from other scene lighting.

Examples

<these are sandbox edits to the whole shading intro section of the wiki, which groups materials and textures, and gives us an entree into Volumetric shading. Note qualification of Mesh object. Need to investigate shading of other object types...>

Shading is the process and the code which enables an object to be seen in the final render output. Blender has four methods to shade a mesh object:

1. Surface
2. Volumetric
3. Halo
4. Wire

Surface shading indicates that the object is a tangible, skinned object that has a solid (but possibly pliable) surface, such as a chair, a sword, or a peach. The surface is described in terms of having a diffuse, specular, mirror, and transparency. It may also have a semi-transparent surface and something inside of it that scatters light, called sub-surface scattering. It may be reflective, such as chrome, smooth plastic, or metal, and may be partially transparent, such as glass, or liquid.

Volumetric shading treats the object as a volume of space that is filled with microscopic particles, such as a cloud, smoke, mist, fog, mystical spells, and steam. As light enters the volume, it is scattered by these particles, and some of that scattering reaches the eye/camera for us to see. The volume is described in terms of density, xxx. The particles may be uniformly colored but have a varying density within the volume, and so the shape may have darker areas. The density may be uniformly dispersed throughout the volume, or it may be clumpled, giving a recognizable shape. Those microscopic particles may give off light themselves, as if they contained glowing embers or sparks, or were transmitting some energy field inside the cloud. That density may be driven by a particle system to create a well-defined jet or emission.

Halo shading turns each vertex of the object into a glob of light, an effect seen with sparks, pixie dust, glint, and sparkles from, for example, a diamond in bright sunlight. Halos can also be used to give a rough approximation of a lens flare, which is observed when a real camera lens looks directly at a bright light source such as the sun.

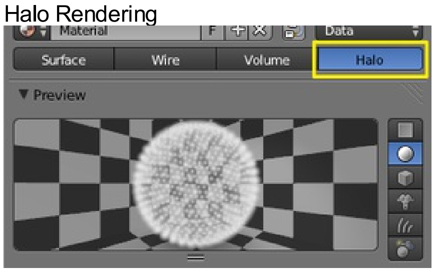
Wire shading renders each edge of the object as a thin line, like a wire cage, or net. Wire rendering is very fast and can be used as a proxy material for a more complicated surface to save time during intermediate renders.

There are two major components to shading: the Material and its Textures. The color that you see is a function of the light and the shading, so you need to also check out the lighting section as well. There are five types of objects in Blender that can be shaded: Mesh, Curve, Surface, Meta, and Text. The table below indicates which types of shading are available for each kind of object. Keep in mind that all types of non-mesh objects can be converted from their type to a Mesh, so, ultimately, all kinds of shading are available for all kinds of objects

Shading available per Object type

	Surface	Halo	Wire	Volumetric
Mesh	yes	full	yes	yes
Curve	if cyclic or extruded	no	no	
Surface	yes	no	yes	
Meta	yes	no	no	
Text	yes	no	no	

Halo Rendering

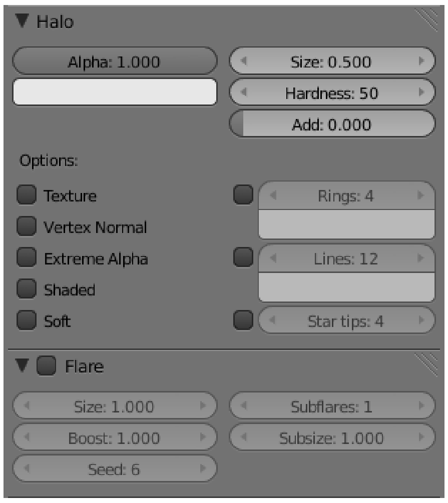


Activating helo rendering

Blender provides a set of materials which do not obey the face-shader paradigm and which are applied on a per-vertex rather than on a per-face basis. These are called Halos because you can see them, but they do not have any substance. They are like little clouds of light; although they are not really lights because they do not cast light into the scene like a lamp.

Halos come in very handy when creating certain special effects, when making an object glow, or when creating a viewable light or fog/atmospherics around an actual light.

Options



Halo panels

To enable Halos, press the Halo button in the Material menu's top panel.

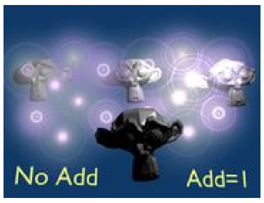
As you will see in the 3D View, the mesh faces are no longer rendered. Instead just the vertex is rendered, since that is where each halo will originate. Halos can be hard to find in a crowded scene, so name it well for easy location in the outliner.

In the properties window, where we normally find the Diffuse, Specular, and Shading panels, we now see panels relative to the Halo characteristics:

Halo Panel

Alpha
The transparency
Color Swatch
The color of the halo itself
Seed
If non-zero, randomizes the ring dimension and line location. To use, give any (integer) number to start the random-number generator.

Size
Sets the dimension of the halo
Hardness
Sets the hardness of the halo. Similar to specular hardness



Effect of Add

Add
The Add slider determine how much the halo colors are 'added to', rather than mixed with, the colors of the objects behind and together with other halos. By increasing Add, the Halo will appear to light up objects that move behind it or through the Halo field.

Texture
: Gives halo a texture. By default, textures are applied to objects with Object coordinates and reflects on the halos by affecting their color, as a whole, on the basis of the color of the vertex originating the halo. Enable this feature to have the texture take effect *within* the halo, and hence to have it with varying colors or transparencies; this will map the whole texture to *every* halo. This technique proves very useful when you want to create a realistic rain effect using particle systems, or similar.

Vertex Normal
: Use the vertex normal to specify the dimension of the halo

Extreme Alpha
: Boosts alpha

Shaded
: Lets halo receive light and shadows from external objects

> When shaded is enabled, the Halo will be affected by local light; a lamp will make it brighter and affect its diffuse color and intensity.

Soft
: Softens the edges of the halos at intersections with other geometry

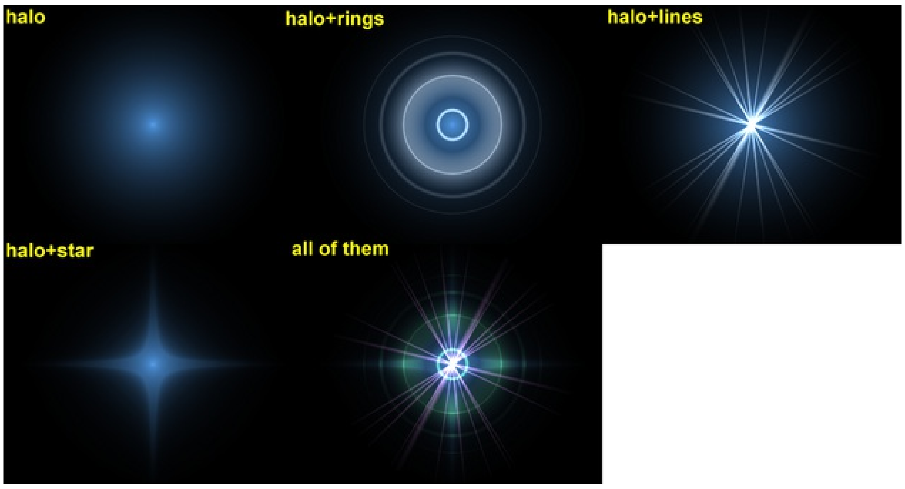
In addition, several other special effects are available. To enable some or all of these effects, set the number of points/rings, or set the color of each effect individually:

Rings
: Adds circular rings around to the halo.

Lines
: Adds lines from the center of the halo.

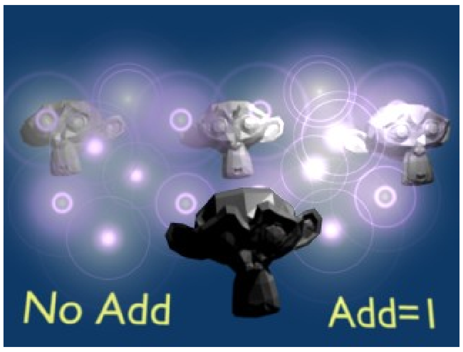
Star tips
: Gives the halo a star shape.

You can not use color ramps. Lines, Rings and an assortment of special effects are available with the relevant toggle buttons, which include Flare, Rings, Lines, Star, Texture, Extreme Alpha, and Shaded. *Halo Variations* shows the result of applying a halo material to a single vertex mesh.



Halo Variations

The halo size, hardness and alpha can be adjusted with the pertinent sliders. These are very similar to traditional material settings



The Add slider determine how much the halo colors are 'added to', rather than mixed with, the colors of the objects behind and together with other halos. By increasing Add, the Halo will appear to light up objects that move behind it or through the Halo field.

To set the number of rings, lines, and star points independently, once they are enabled with the relative Toggle Button, use the Num Buttons Rings:, Lines: and Star:. Rings and lines are randomly placed and oriented, to change their pattern you can change the Seed: Num Button which sets the random numbers generator seed.

Flare Panel

Enabling Flare Renders the halo as a lens flare

Size
: Sets the factor by which the flare is larger than the halo

Boost
: Give the flare extra strength.

Seed
: Specifies an offset in the flare seed table

Subflares
: Sets the number of subflares

Subsize
: Sets the dimensions of the subflares, dots, and circles

Lens Flares

Our eyes have been trained to believe that an image is real if it shows artifacts that result from the mechanical process of photography. *Motion blur*, *Depth of Field*, and *lens flares* are just three examples of these artifacts. The first two are discussed in the *chapter_rendering*; the latter can be produced with special halos. A simulated lens flare tells the viewer that the image was created with a camera, which makes the viewer think that it is authentic.

We create lens flares in Blender from a mesh object using first the Halo button and then the Flare options in the Shaders Panel of the material settings. Try turning on Rings and Lines, but keep the colors for these settings fairly subtle. Play with the Flares: number and Fl.seed: settings until you arrive at something that is pleasing to the eye. You might need to play with Boost: for a stronger effect (*Lens Flare settings*).

Note that this tool does not simulate the physics of photons traveling through a glass lens; it's just a eye candy.

Blender's lens flare looks nice in motion, and disappears when another object occludes the flare mesh.

Lens Flare

Halo Texturing

By default, textures are applied to objects with Object coordinates and reflects on the halos by affecting their color, as a whole, on the basis of the color of the vertex originating the halo. To have the texture take effect *within* the halo, and hence to have it with varying colors or transparencies press the Texture button; this will map the whole texture to *every* halo. This technique proves very useful when you want to create a realistic rain effect using particle systems, or similar.

Another Option is Shaded. When shaded is enabled, the Halo will be affect by local light; a lamp will make it brighter and affect its diffuse color and intensity.

Examples

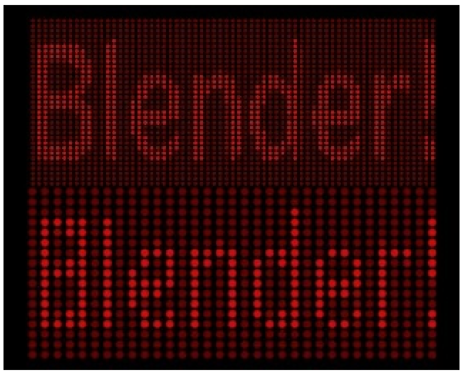
Dotmatrix display

Let's use a halo material to create a dotmatrix display.

- To begin, add a grid with the dimensions 32x16. Then add a camera and adjust your scene so that you have a nice view of the billboard.
- Use a 2D image program to create some red text on a black background, using a simple and bold font (if you are a lazy lizard [I hope this not offensive, I just like how it sounds!], you can just save the picture below on your hard drive…). *Dot matrix image texture.* shows an image 512 pixels wide by 64 pixels high, with some black space at both sides.

Dot matrix image texture.

- Add a material for the billboard, and set it to the type Halo. Set the HaloSize to 0.06 and when you render the scene you should see a grid of white spots.
- Add a Texture, then change to the Texture Buttons and make it an image texture. When you load your picture and render again you should see some red tinted dots in the grid.
- Return to the Material Buttons and adjust the sizeX parameter to about 0.5 then render again; the text should now be centered on the Billboard.
- To remove the white dots, adjust the material color to a dark red and render. You should now have only red dots, but the billboard is still too dark. To fix this enter EditMode for the board and copy all vertices using the ⇧ ShiftD shortcut (take care not to move them!). Then adjust the brightness with the Add value in the MaterialButtons.


Dot Matrix display.

You can now animate the texture to move over the billboard, using the ofsX value in the Texture panel of the MaterialButtons. (You could use a higher resolution for the grid, but if you do you will have to adjust the size of the halos by shrinking them, or they will overlap. (*Dot Matrix display*).

Note about material indices
Halo materials only work when applied using the first material index. Any material(s) in a subsequent material index will not be rendered.

Introduction to Textures

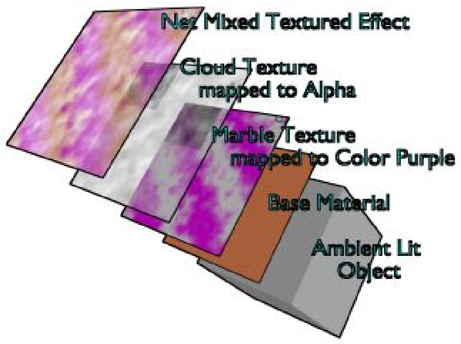
In CGI, texture mapping is a method to add detail to surfaces by projecting images and patterns onto those surfaces. The projected images and patterns can be set to affect not only color, but also specularity, reflection, transparency, and even fake 3-dimensional depth. Most often, the images and patterns are projected during render time, but texture mapping is also used to sculpt, paint and deform objects.

In Blender, Textures can be:

- applied to a *Material*
- applied to a light, that coming from lamp
- applied to the World Background
- applied to a *Brush*, see for example:
  - Sculpt Mode
  - Painting the Texture
- associated with Modifiers, see:
  - Particles textures
  - Ocean textures

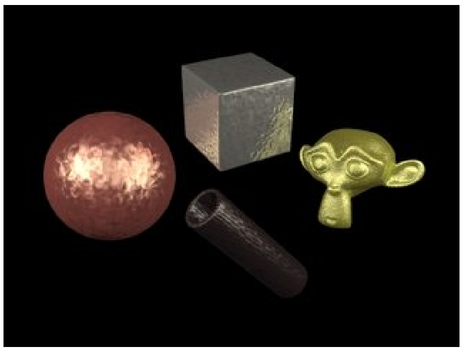
Material Textures

The material settings that we've seen so far produce smooth, *uniform* objects, but such objects aren't particularly true to reality, where uniformity tends to be uncommon and out of place. In order to deal with this unrealistic uniformity, Blender allows the user to apply *textures* which can modify the reflectivity, specularity, roughness and other surface qualities of a material.


Net Mixed Textured Effect
Cloud Texture mapped to Alpha
Marble Texture mapped to Color Purple
Base Material
Ambient Lit Object
Textures Layer on base Material

Textures are like additional layers on top of the base material. Textures affect one or more aspects of the object's net coloring. The net color you see is a sort of layering of effects, as shown in this example image. The layers, if you will, are:

1. Your object, lit with **ambient** light based on your world settings.
2. Your base **material**, which colors the whole surface in a uniform color that reacts to light, giving different shades of the diffuse, specular, and mirror colors based on the way light passes through and into the surface of the object.
3. A **primary texture** layer that overlays a purple marble coloring.
4. A **second cloud texture** that makes the surface transparent in a misty/foggy sort of way by affecting the Alpha value.
5. These two textures are **mixed** with the base material to provide the net effect: a cube of purplish-brown fog.


Some Metal Textures

This notion of using *more than one* texture, to achieve a combined effect, is one of the "hidden secrets" of creating realistic-looking objects. If you carefully "look at the light" while examining any real-life object, you will observe that the final appearance of that object is best described as the combination, in different ways and in different amounts, of several distinct underlying visual characteristics. These characteristics might be more (or less) strongly apparent at different angles, under different lighting conditions, and so forth. Blender allows you to achieve this in many ways. You can use "a stack of texture layers" as described in this section, or you can also use arbitrarily-complex networks ("noodles"...) of "texture nodes" as discussed here; the choice is yours.

Materials Textures fall into three primary categories:

Procedural Textures
: Textures generated by a mathematical formula. For example, Wood, Clouds, and Distorted Noise

Images or Movies
: Photos and films projected onto objects. For example, a flat map of Earth mapped to a sphere.

Environment Maps
: Textures used to create the impression of reflections and refractions. For example, an image of a street reflected in a car window.

Data or Modifiers Textures
: Textures obtained from raw data or obtained by a certain modifier in the scene.
For example:

- volumetric materials use Voxel Data textures, or Point Density textures
- textures can be obtained from an Ocean Modifier

CRL 02:25, 26 May 2014 (UTC)(Sign)

World Textures

in progress ▭ 10% just started

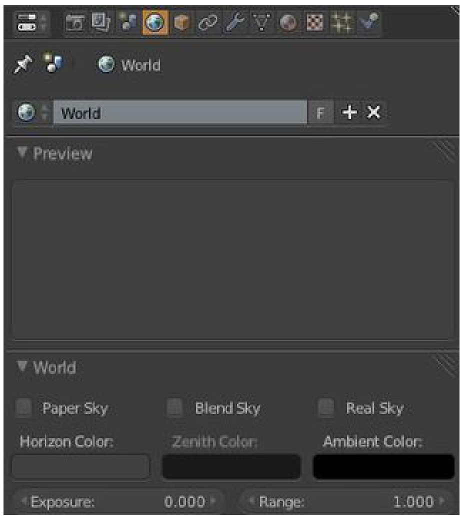
Mode: All Modes

Panel: Shading/World Context → Preview

Hotkey:

Description

The world buttons let you set up the shading of your scene in general. It can provide ambient colour, and special effects such as mist, but a very common use of a World is to shade a background colour.



Textures Layer on base Material

HoR, HoG, HoB
: The RGB color at the horizon

ZeR, ZeG, ZeB
: The RGB color at the zenith (overhead)

These colors are interpreted differently, according to the Buttons in the Preview Panel (*Background colors*):

None Enabled
: If none of these three buttons is checked, your background will just be plain flat color (using the horizon one).

Blend
: The background color is blended from horizon to zenith. If only this button is pressed, the gradient runs from the bottom to the top of the rendered image regardless of the camera orientation.

Real
: If this option is added, the gradient produced has two transitions, from nadir (same color as zenith) to horizon to zenith; the blending is also dependent on the camera orientation, which makes it more realistic. The horizon color is exactly at the horizon (on the x-y plane), and the zenith color is used for points vertically above and below the camera.

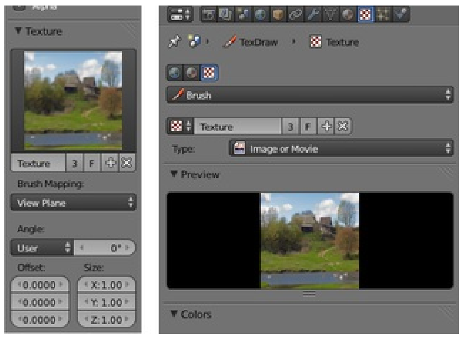
Paper
: If this option is added, the gradient keeps its characteristics, but it is clipped in the image (it stays on a horizontal plane (parallel to x-y plane): what ever the angle of the camera may be, the horizon is always at the middle of the image).

CRL 02:31, 26 May 2014 (UTC)(Sign)

Brush Textures

in progress ▭ 10% just started

Image textures can be loaded into blender. These images can then be applied to a mesh model that has been unwrapped and assigned an image of user defined size.



Texture 3 F

Brush Mapping: View Plane

Angle: User 0°

Offset: 0.0000 0.0000 0.0000

Size: X: 1.00 Y: 1.00 Z: 1.00

Texture 3 F

Type: Image or Movie

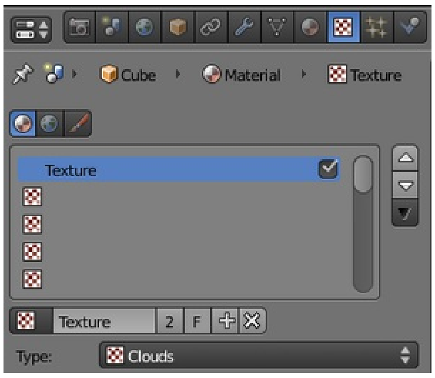
Applied Brush texture in different painting modes

- Brush textures can be used to paint textures.
- Brush textures can be used to paint vertices.
- Brush textures can also be used in sculpting to create topology.

Assigning a Texture

This page just shows how to add a texture to a slot. The textures' commons options are explained here.

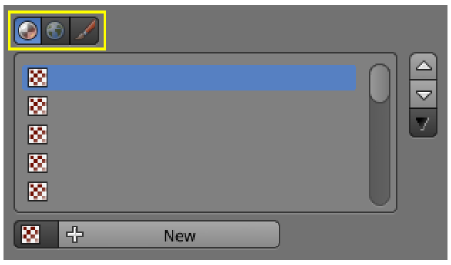
Choosing the Texture context



Texture panel

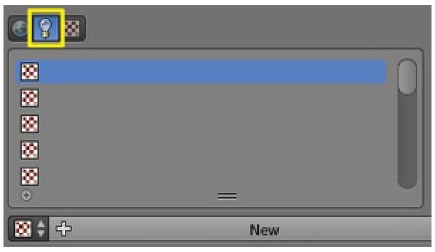
In the Properties editor, choose the Texture context: this will show the Texture panel.

Choosing the Texture data type



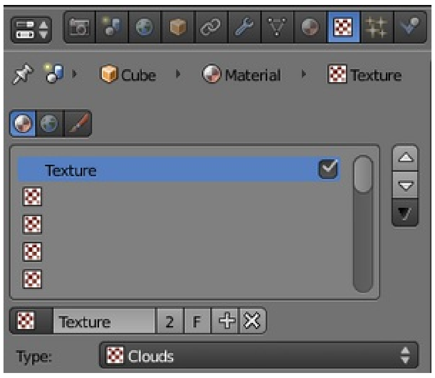
Texture panel with buttons for Material, World, and Brush textures highlighted

The three buttons Material, World, Brush at the top of the texture panel indicate the texture data type, that is, the kind of texture that is being edited.



Texture panel with button for Lamp textures highlighted

Textures Slots



Texture panel

The list below these buttons represent the Stack of textures that we can manage. It can have up to eighteen Texture Slots:

- Tick or untick a texture to enable/disable it.
- Use the three buttons on the right side to move individual textures up and down in the stack or to copy/paste material's settings between slots.

Creating a new Texture Datablock in a new Texture Slot

Select an empty slot, then click on the + New button.

This will do two things:

- it will create a new texture datablock
- also, it will add a new slot in the textures stack

Creating a new Texture Datablock in a non-empty slot

Select a non-empty slot, then click on the + button.

This will do two things:

- it will create a new texture datablock, with a new name, **making a copy of the texture datablock assigned to the selected slot**
- it will assign this new datablock to the selected slot

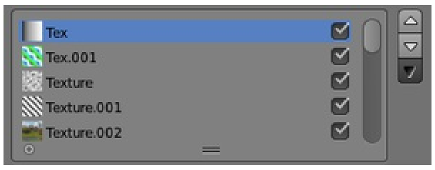
Sharing a Texture Datablock in a non-empty slot

- Select a non-empty slot, then click on the Browse button. This will open a menu showing all the available Texture Datablocks in this file.
- Choose a texture datablock in the menu to assign it to the selected slot. This will share the chosen texture with more than one object, hence the *Number of users* shown in the texture datablock will increase by one.

Textures common options

In the Properties editor, choose the Texture context: this will show the Texture panel.

Textures Stack



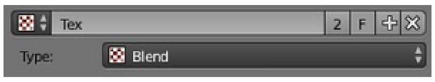
Textures Stack

The list below these buttons represents the Stack of textures that we can manage. It can have up to eighteen Texture Slots:

- Tick or untick a texture to enable/disable it.
- Use the three buttons on the right side to move individual textures up and down in the stack or to copy/paste material's settings between slots.

The order in the Stack Textures defines how textures overlay each other for finally result image.

Texture Datablock



Active Texture Datablock

Select a slot in the Textures Stack to see its settings.

The first group of buttons below the stack displays the texture currently selected in the stack.

Browse
: The first button below the stack displays the all available textures in the current file. Textures are stored globally, and can be linked to more than one material. If you have already created a texture that you want to reuse, select from this list.

Name
: A name field where the name of the material can be changed.

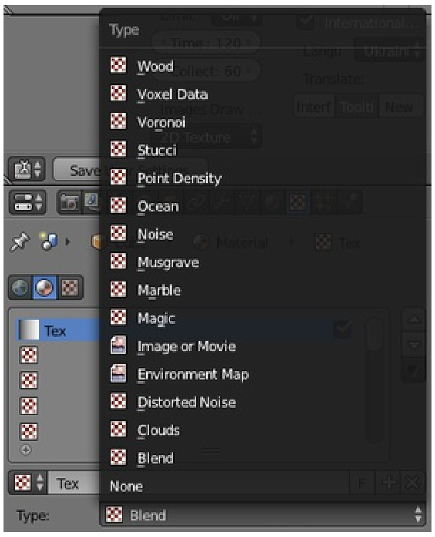
Number of users
: If the active texture is used by another material, a 2 button appears that can be used to make a single-user copy of the active texture. Use this button to quickly create a new texture based on an existing texture.

Fake
: The F button assigns the active texture to a "Fake" material, so that the texture is saved with the file even if it has no "real" users.

Add
: Replaces the texture of the active slot with a new texture.

Unlink
: Removes the texture from the active slot.

Texture Type



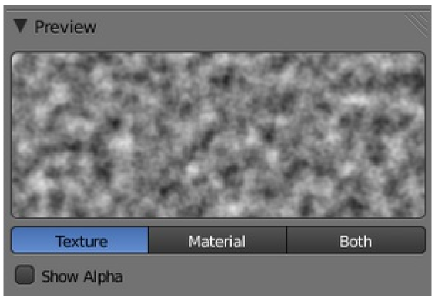
Texture Types

Choose the type of texture that is used for the current texture datablock.

- Procedural Textures
- Image and Video Textures
- Environment Map
- Volume Textures
- Ocean Texture

These types are described in detail in this section.

Preview



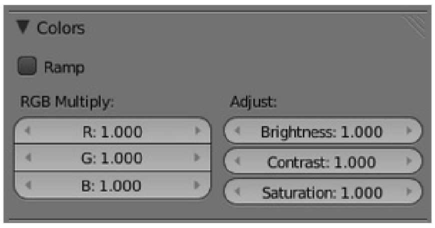
Preview panel

The texture preview panel provides a quick pre-visualisation of how the texture looks on its own, without mapping.

Texture, Material, or Both
: Choose to display only the texture, only the material, or both.

Show Alpha
: Show alpha in preview.
If Alpha: Use is checked in the Image Sampling panel, the image's alpha channel is displayed.
If Alpha: Use is unchecked, an alpha channel based on averaged rgb values is displayed like it would be used by the Alpha slider in the Influence panel.

Colors



R: 1.000
G: 1.000
B: 1.000
Brightness: 1.000
Contrast: 1.000
Saturation: 1.000

Colors panel

The Ramp button activates a color ramp which allows you to remap the colors of a texture to new ones. See Ramps for information on using ramps.

The color of a texture can be modified with the Brightness, Contrast, and Saturation buttons. All textures with RGB-Values — including Images and Environment Maps — may be modified with the RGB sliders.

R, G, B
: Tint the color of a texture by brightening each red, green and blue channel.

Brightness
: Change the overall brightness/intensity of the texture

Contrast
: Change the contrast of the texture

Saturation
: Change the saturation of the texture

Mapping

Here you can control how the texture will be mapped on the object.

Brushes
These options are not available for brushes because they wouldn't make sense

See Mapping section for details.

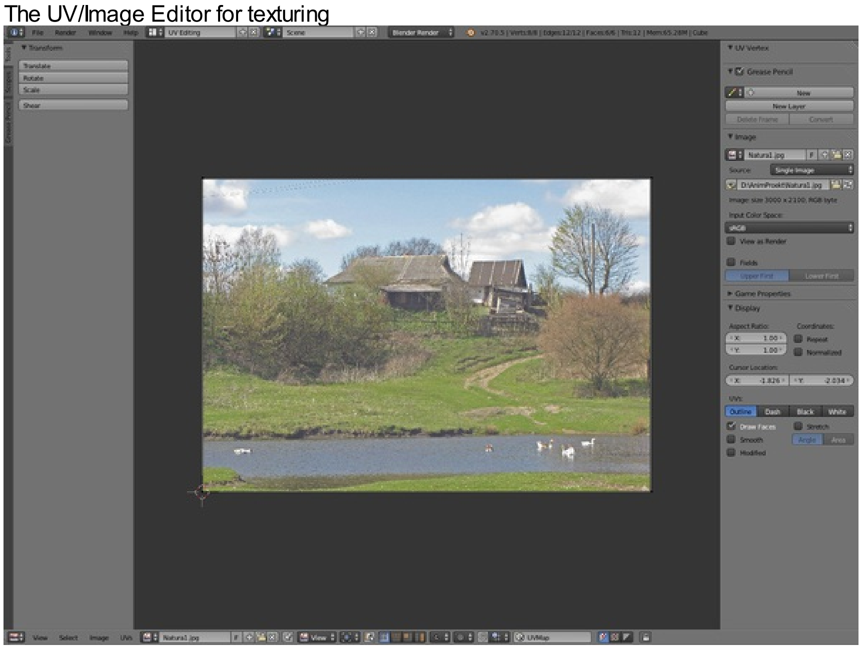
Influence

Here you can control what properties the texture will affect, and by how much.

They are detailed on the Influence section.

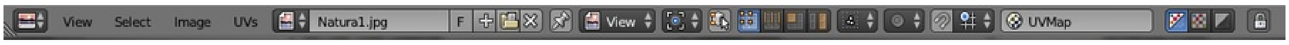
Brushes
These options are not available for brushes because they wouldn't make sense

The UV/Image Editor for texturing


UV/Image Editor window for texturing

The UV/Image Editor is where you will be editing the UVs. This is an overview of the tools found there. Using the UV editor is explained more in depth in the next sections.

Header Bar



UV/Image Editor Header

The header bar contains several menus and options for working with UVs

View menu
Tools for Navigating, working with the editor and controlling how things are displayed. The properties panel has display options and manipulation tools. When an image is being used, image properties are displayed. The Scopes panel is used when working with Images. It contains different image visualizers

Select menu
Tools for Selecting UVs.

Image menu
This contains options for when Working with Images and Painting Textures.

UVs menu
Contains tools for Unwrapping Meshes and Editing Uvs.

Image Selector Menu
Select the image to apply when Working with Images.

Pin Image
Displays current image regardless of selected object.

Pivot Point Selector
Similar to working with Pivot Points in the 3D view.

Sync Selection
Keeps UV and Mesh component selections in sync.

Selection Modes

- Vertex
- Edge
- Face
- Island

Sticky Selection Mode
When Sync Selection is disabled, these options control how UVs are selected.

Proportional Editing
Works like Proportional Editing in the 3d view

UV Snapping
Similar to Snapping in the 3D View

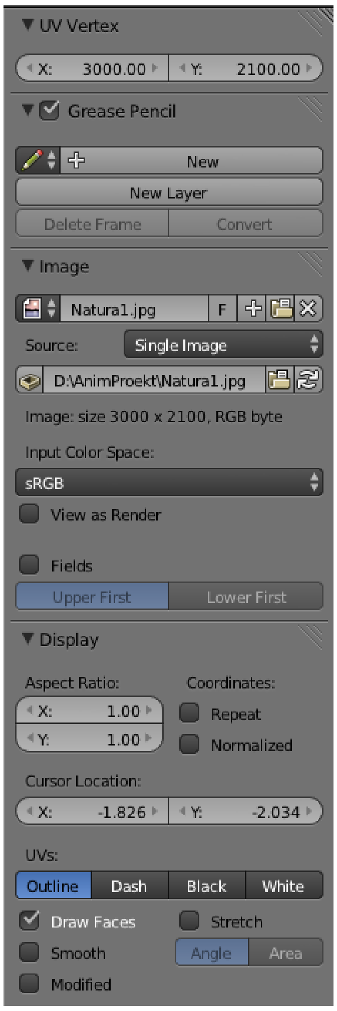
Active UV Texture Map Selector
Select which UV texture to use

Image Channels to Draw

Set the image to be displayed with Color, Color and Alpha, or just Alpha.

Auto Update Other Affected Windows
: Update other affected windows space automatically to reflect changes during interactive operations such as transfom.

Properties Panel



UV/Image Editor Properties panel

UV Vertex
: Transform Properties for select UVs

Grease Pencil
: Similar to Grease Pencil in the 3d view.

Image
: Contains the properties of the current Image

Display
: Controls display options for UVs and additional settings for when Working with Images.

Display Options

You can set how UVs are displayed in the Display Panel:

Aspect Ratio
: Display Aspect for this image. Does not affect rendering.

Coordinates
: Display UV coordinates

  Repeat
  : Draw the image repeated outside of the main view.

  Normalized
  : Display UV coordinates from 0.0 to 1.0 rather then in pixels

Cursor Location
: 2D cursor location for this view

Outline/Dash/Black/White
: Sets how UV edges are displayed

Draw Faces
: Draw faces over the image

Smooth
: Makes edges appeared Antialiased

Modified
: Show results of modifiers in the UV display

Stretch
: Shows how much of a difference there is between UV coordinates and 3D coordinates. Blue means low distortion, while Red means high distortion. Choose to display the distortion of Angles or the Area.

Navigating in UV Space

Panning can be done by clicking the MMB and dragging.

Zooming can be done by scrolling MMB up or down. Also, as in the 3D view, you can use + NumPad or - NumPad to zoom.

The following shortcuts are available, and through the View Menu:

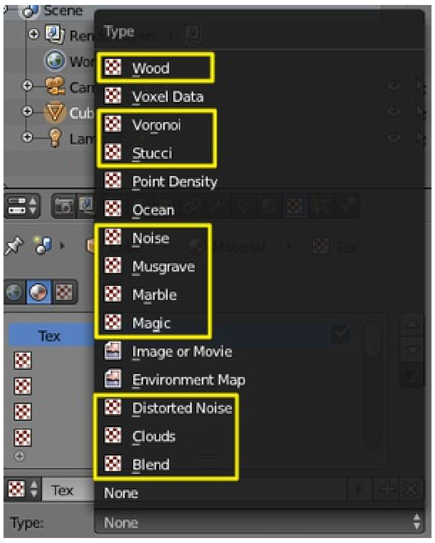
- Zoom 1:8 8 NumPad
- Zoom 1:4 4 NumPad
- Zoom 1:2 2 NumPad
- Zoom 1:1 1 NumPad
- Zoom 2:1 ⇧ Shift2 NumPad
- Zoom 4:1 ⇧ Shift4 NumPad
- Zoom 8:1 ⇧ Shift8 NumPad

- View All ↖ Home
- View Center . NumPad

Texture types

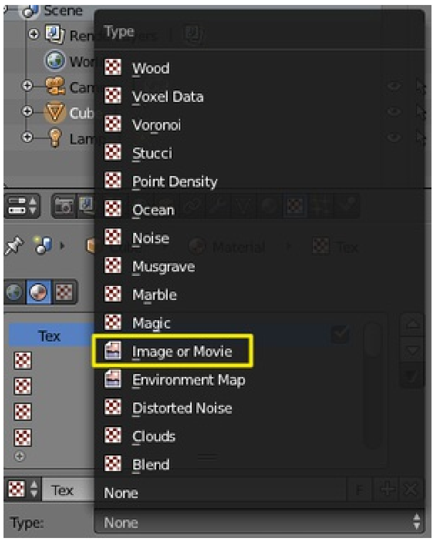
This are the available texture types:

- Procedural Textures



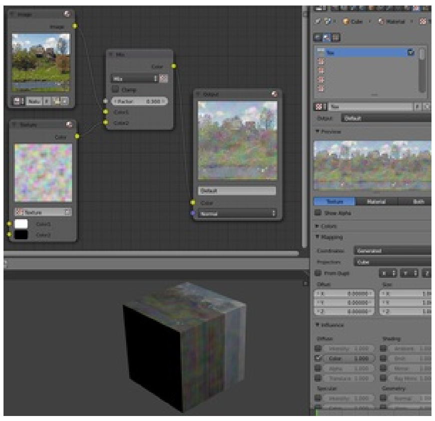
Textures generated by a mathematical formula.

- Image Textures
- Video Textures



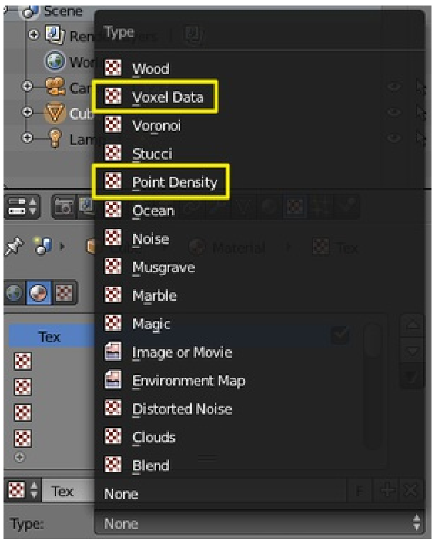
Photos and films projected onto objects.

- Combined Textures



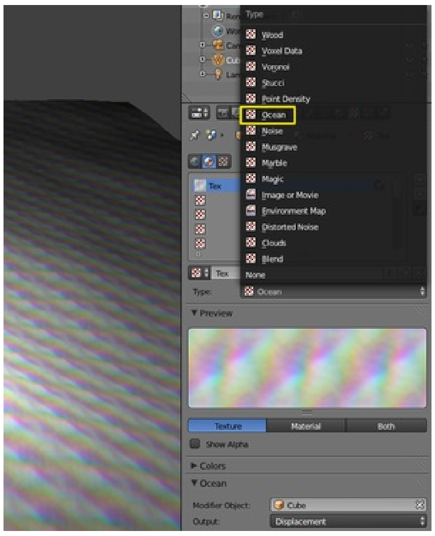
Combined textures based on nodes.

- Volume Textures



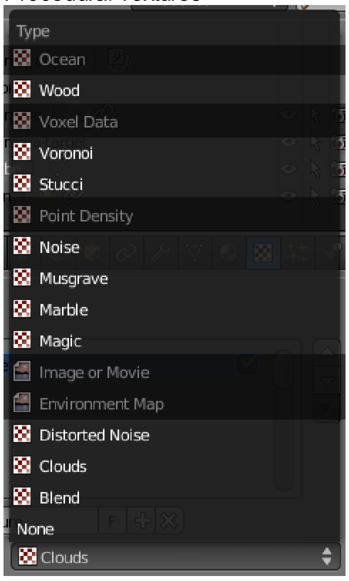
Textures that can be applied to volumetric data.

- Ocean Textures



Texture generated by an Ocean modifier.

Procedural Textures



The Texture Type list in the Texture panel of the Texture Buttons. (Non procedural textures darkened out.)

Procedural textures are textures that are defined mathematically. They are generally relatively simple to use, because they don't need to be mapped in a special way - which doesn't mean that procedural textures can't become very complex.

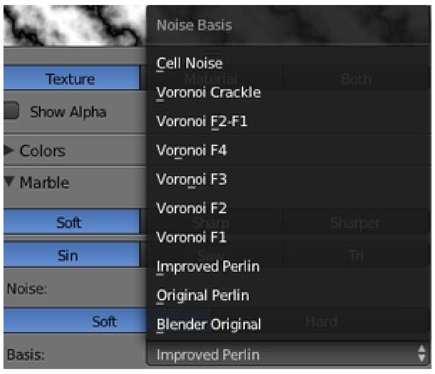
These types of textures are 'real' 3D. By that we mean that they fit together perfectly at the edges and continue to look like what they are meant to look like even when they are cut; as if a block of wood had really been cut in two. Procedural textures are not filtered or anti-aliased. This is hardly ever a problem: the user can easily keep the specified frequencies within acceptable limits.

These are the available types:

- Blend
- Clouds
- Distorted Noise
- Magic
- Marble
- Musgrave
- Noise
- Stucci
- Voronoi
- Wood

Common options

Noise Basis



Noise Basis list

Each noise-based Blender texture (with the exception of Voronoi and simple noise) has a Noise Basis setting that allows the user to select which algorithm is used to generate the texture. This list includes the original Blender noise algorithm. The Noise Basis settings makes the procedural textures extremely flexible (especially Musgrave).
The Noise Basis governs the structural appearance of the texture :

Blender Original	Voronoi F1	Voronoi F2-F1
Original Perlin	Voronoi F2	Voronoi Crackle
Improved Perlin	Voronoi F3	Cell Noise
	Voronoi F4	

There are two more possible settings for Noise Basis, which are relatively similar to Blender Original: Improved Perlin and Original Perlin

Nabla

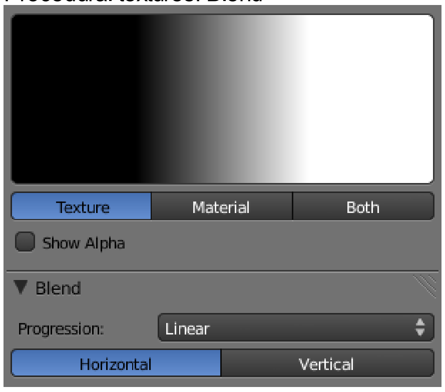
Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (with as exception Blend and Magic). This is important for Normal and Displacment Maps. The strength of the effect is controlled with the Nabla Number Button.

Hints

Use the size buttons in the Mapping panel to set the size that the procedural textures are mapped to.

Procedural textures can either produce colored textures, intensity only textures, textures with alpha values and normal textures. If intensity only ones are used the result is a black and white texture, which can be greatly enhanced by the use of ramps. If on the other hand you use ramps and need an intensity value, you have to switch on No RGB in the Mapping panel.

Procedural textures: Blend



Progression: Linear
Horizontal
Vertical

Blend Texture Panels

Often used for
: This is one of the most frequently used procedural textures. You can use blend textures to blend other textures together (with Stencil), or to create nice effects (especially with the Mapping: Normal trick). Just remember: if you use a ramp to create a custom blending, you may have to use No RGB, if the Mapping value needs an intensity input.

Result(s)
: Intensity. The Blend texture generates a smoothly interpolated progression.

Options

Progression
: Profile of blend

  Linear
  : A linear progression

Quadratic
: A quadratic progression

Easing
: A flowing, non-linear progression

Diagonal
: A diagonal progression

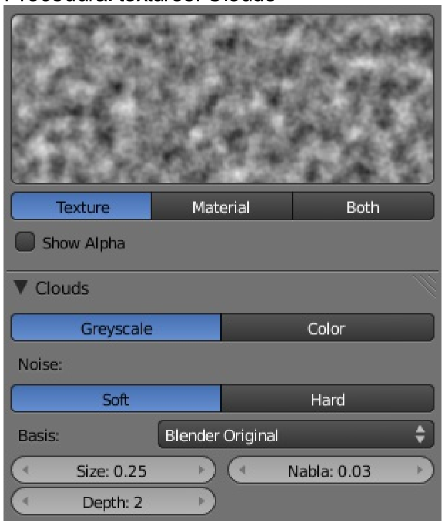
Spherical
: A progression with the shape of a three-dimensional ball

Quadratic Sphere
: A quadratic progression with the shape of a three-dimensional ball

Radial
: A radial progression

Horizontal/Vertical
: The direction of the progression is flipped a quarter turn.

Procedural textures: Clouds



Clouds Texture Panels

Clouds represent Perlin noise. In addition, each noise-based Blender texture (with the exception of Voronoi and simple noise) has a "Noise Basis" setting that allows the user to select which algorithm is used to generate the texture.

Often used for
: Clouds, Fire, Smoke. Well-suited to be used as a Bump map, giving an overall irregularity to the material.

Result(s)
: Greyscale (default) or RGB Color

Options

Greyscale
: The standard noise, gives an intensity

Color
: The noise gives an RGB value

Noise
: Soft or Hard, changes contrast and sharpness

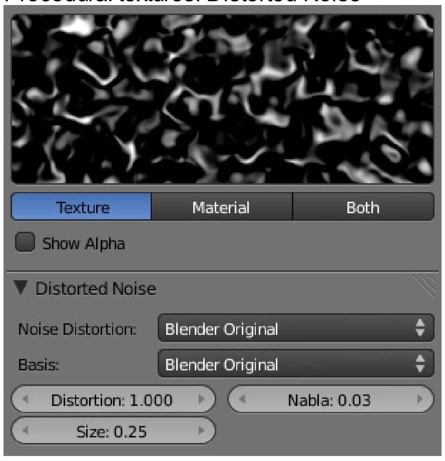
Size
: The dimension of the Noise table

Depth
: The depth of the Clouds calculation. A higher number results in a long calculation time, but also in finer details.

Technical Details

A three-dimensional table with pseudo-random values is used, from which a fluent interpolation value can be calculated with each 3D coordinate (thanks to Ken Perlin for his masterful article "An Image Synthesizer", from the SIGGRAPH proceedings 1985). This calculation method is also called Perlin Noise.

Procedural textures: Distorted Noise



Texture | Material | Both

Show Alpha

Distorted Noise

Noise Distortion: Blender Original

Basis: Blender Original

Distortion: 1.000 Nabla: 0.03

Size: 0.25

Distorted Noise Texture Panels

Distortion Noise takes the option that you pick from Noise Basis and filters it, to create hybrid pattern.

Often used for
: Grunge, very complex and versatile

Result(s)
: Intensity

Options

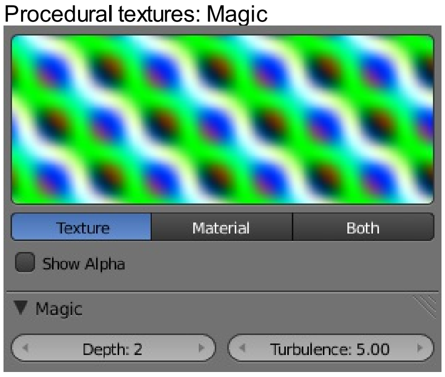
Noise Distortion
: The texture to use to distort another

Basis
: The texture to be distorted

Noise
: The size of the noise generated

Distortion
: The amount that Distortion Noise affects Basis

Procedural textures: Magic



Magic Texture Panels

Often used for
: Not frequently used. It can be used for "Thin Film Interference", if you set Mapping to Reflection and use a relatively high Turbulence.

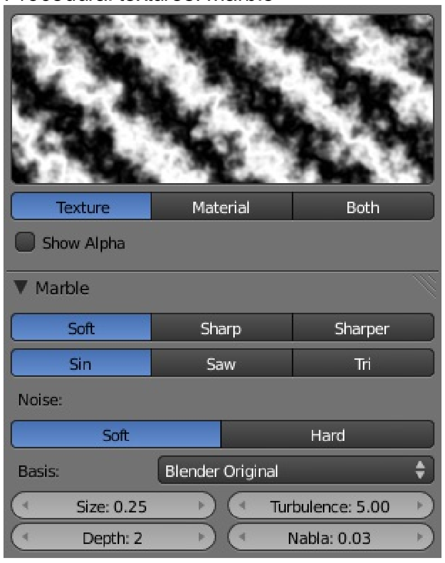
Result(s)
: RGB color. The RGB components are generated independently with a sine formula.

Options

Depth
: The depth of the calculation. A higher number results in a long calculation time, but also in finer details.

Turbulence
: The strength of the pattern.

Procedural textures: Marble



Marble Texture Panels

Often used for
: Marble, Fire, Noise with a structure

Result(s)
: Intensity value only

Bands are generated based on the sine, saw, or triangular formulae and noise turbulence.

Options

Soft/Sharp/Sharper
: Three presets for soft to more clearly defined Marble

Sin/Saw/Tri
: Shape of wave to produce bands

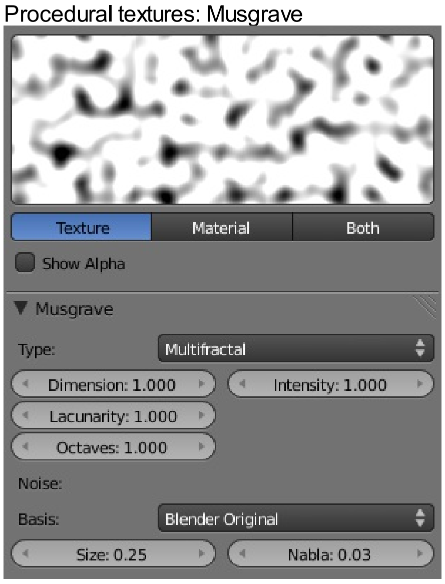
Soft/Hard
: The noise function works with two methods.

Size
: The dimensions of the noise table

Depth
: The depth of the Marble calculation. A higher value results in greater calculation time, but also in finer details.

Turbulence
: The turbulence of the sine bands.

Procedural textures: Musgrave



Type: Multifractal
Dimension: 1.000
Intensity: 1.000
Lacunarity: 1.000
Octaves: 1.000
Basis: Blender Original
Size: 0.25
Nabla: 0.03

Musgrave Texture Panels

Often used for
: Organic materials, but it's very flexible. You can do nearly everything with it.

Result(s)
: Intensity

Options

Type
: This procedural texture has five noise types on which the resulting pattern can be based and they are selectable from a dropdown menu at the top of the tab. The five types are:

- Hetero Terrain
- fBm
- Hybrid Multifractal
- Ridged Multifractal
- Multifractal

These noise types determine the manner in which Blender layers successive copies of the same pattern on top of each other at varying contrasts and scales.

Examples with Basis : Voronoi F1 - Dimension : 0.5 - Lacunarity : 0.15 - Octave: 2.0

Hetero Terrain fBM Hybrid Multifrct Ridged Multifrct Multifractal

The main noise types have four characteristics:

Dimension
: Fractal dimension controls the contrast of a layer relative to the previous layer in the texture. The higher the fractal dimension, the higher the contrast between each layer, and thus the more detail shows in the texture. Range: 0 to 2.

Lacunarity
: Lacunarity controls the scaling of each layer of the Musgrave texture, meaning that each additional layer will have a scale that is the inverse of the value which shows on the button. i.e. Lacunarity = 2 -> Scale = 1/2 original. Range: 0 to 6.

Octaves
: Octave controls the number of times the original noise pattern is overlayed on itself and scaled/contrasted with the fractal dimension and lacunarity settings. Range: 0 to 8.

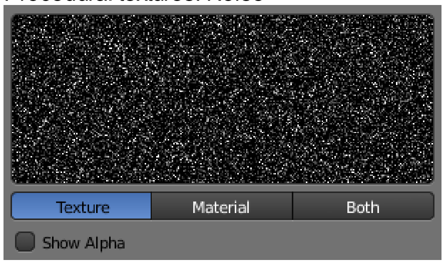
Intensity
: Light intensity. Called Offset for Hetero Terrain. Range: 0 to 10.

The Hybrid Multifractal and Ridged Multifractal types have these additional settings:

Offset
: Both have a "Fractal Offset" button that serves as a "sea level" adjustment and indicates the base height of the resulting bump map. Bump values below this threshold will be returned as zero. Range: 0 to 6.

Gain
: Setting which determines the range of values created by the function. The higher the number, the greater the range. This is a fast way to bring out additional details in a texture where extremes are normally clipped off. Range: 0 to 6.

Procedural textures: Noise



Noise Texture Panel

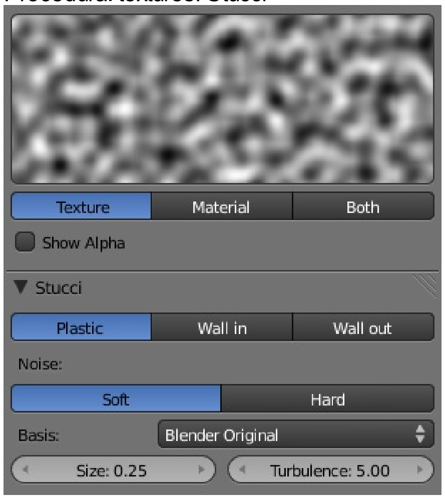
Although this looks great, it is not Perlin Noise! This is a true, randomly generated Noise. This gives a different result every time, for every frame, for every pixel.

There are no options for this noise

Often used for
: White noise in an animation. This is not well suited if you don't want an animation. For material displacement or bump, use clouds instead.

Result(s)
: Intensity

Procedural textures: Stucci



Stucci Texture Panels

The Stucci texture is based on noise functions.

Often used for
: Stone, Asphalt, Oranges. Normally for Bump-Mapping to create grainy surfaces.

Result(s)
: Normals and Intensity

Options

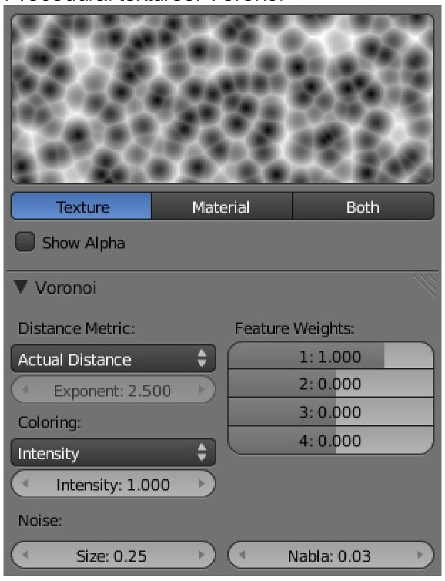
Plastic/Wall In/Wall out
: Plastic is the standard Stucci, whilst the "walls" is where Stucci gets it name. This is a typical wall structure with holes or bumps.

Soft/Hard
: There are two methods available for working with Noise

Size
: Dimension of the Noise table

Turbulence
: Depth of the Stucci calculations

Procedural textures: Voronoi



Distance Metric: Actual Distance
Exponent: 2.500
Coloring: Intensity
Intensity: 1.000
Feature Weights: 1: 1.000, 2: 0.000, 3: 0.000, 4: 0.000
Noise: Size: 0.25, Nabla: 0.03

Voronoi Texture Panels

Often used for
: Very convincing Metal, especially the "Hammered" effect. Organic shaders (e.g. scales, veins in skin).

Result(s)
: Intensity (default) and Color

Options

Distance Metric
: This procedural texture has seven Distance Metric options. These determine the algorithm to find the distance between cells of the texture. These options are:

- Minkovsky
- Minkovsky 4
- Minkovsky 1/2
- Chebychev
- Manhattan
- Distance Squared
- Actual Distance

The Minkovsky setting has a user definable value (the Exponent button) which determines the Minkovsky exponent (e) of the distance function $(x^e + y^e + z^e)^{1/e}$. A value of one produces the Manhattan distance metric, a value less than one produces stars (at **0.5**, it gives a Minkovsky 1/2), and higher values produce square cells (at **4.0**, it gives a Minkovsky 4, at **10.0**, a Chebychev). So nearly all Distance Settings are basically the same - variations of Minkowsky.
You can get irregularly-shaped rounded cells with the Actual Distance/Distance Squared options.

Minkovsky Exponent : 0.5 (Minkovsky 1/2)	Minkovsky Exponent : 1 (Manhattan)	Minkovsky Exponent : 2 (Actual Distance)
Minkovsky Exponent : 4 (Minkovsky 4)	Minkovsky Exponent : 10 (Chebychev)	Distance Squared (More contrast than ActualDistance)

Feature Weights
: These four sliders at the bottom of the Voronoi panel represent the values of the four Worley constants, which are used to

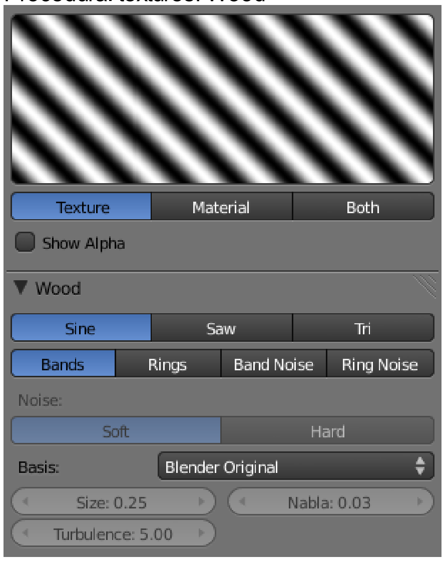
calculate the distances between each cell in the texture based on the distance metric. Adjusting these values can have some interesting effects on the end result.

Coloring
: Four settings (Intensity, Position, Position and Outline, and Position, Outline, and Intensity) that can use four different noise basis as methods to calculate color and intensity of the texture output. This gives the Voronoi texture you create with the "Worley Sliders" a completely different appearance and is the equivalent of the noise basis setting found on the other textures.

Technical Details

For a more in depth description of the Worley algorithm, see: Worley Documentation(dead link).

Procedural textures: Wood



Wood Texture Panels

Often used for
: Woods and ring-shaped patterns.

Result(s)
: Intensity only

Options

Sin/Saw/Tri
: Shape of wave to produce bands

Bands/Rings/Band Noise/Ring Noise
: Set the bands to either straight or ring-shaped, with or without turbulence

Soft/Hard
: There are two methods available for the Noise function

Size
: Dimension of the Noise table

Turbulence
: Turbulence of the Band Noise and Ring Noise types

Technical Details

Generation
: Bands are generated based on a sine formula. You can also add a degree of turbulence with the Noise formula.

Coordinates
: As the band is based on a sine formula, the texture repeats itself every pi units rather than every 1.0 units. To correct this, scale the texture by a value of pi for the dimension you wish.

Image Textures

The term Image Texture simply means that a graphic image — a pixel grid composed of R, G, B, and sometimes Alpha values — is used as the input source to the texture. As with other types of textures, this information can be used in a number of ways, not only as a simple "decal".

When the Texture Type Image or Movie is selected, three new panels present themselves allowing us to control most aspects of how image textures are applied: Image, Image Sampling, and Image Mapping.

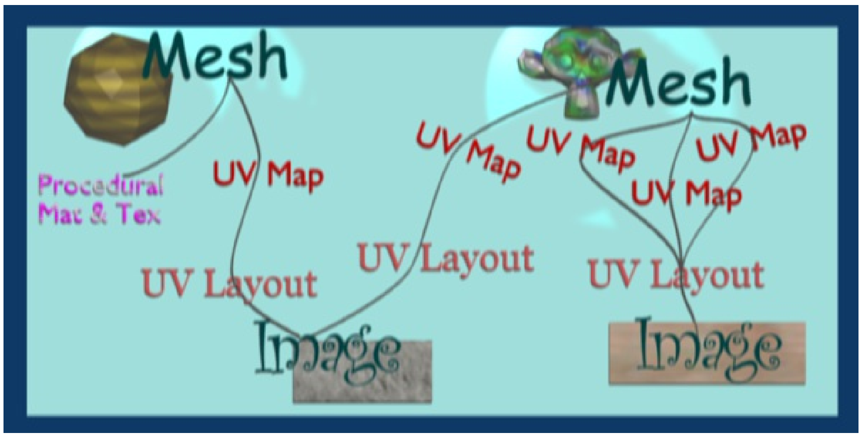
About Image Based Texturing

Texture images take up precious memory space, often being loaded into a special video memory bank that is very fast and very expensive, so it is often very small. So, keep the images as small as possible. A 64x64 image takes up only one fourth the memory of a 128x128 image.

For photo-realistic rendering of objects in animations, often larger image textures are used, because the object might be zoomed in on in camera moves. In general, you want to use a texture sized proportionally to the number of pixels that it will occupy in the final render. Ultimately, you only have a certain amount of physical RAM to hold an image texture and the model and to provide work space when rendering your image.

For the most efficient memory usage, image textures should be square, with dimensions as powers of 2, such as 32x32, 64x64, 128x128, 256x256, 1024x1024, 2048x2048, and 4096x4096.

If you can re-use images across different meshes, this greatly reduces memory requirements. You can re-use images if you map those areas of the meshes that "look alike" to a layout that uses the common image. In the overview below, the left image is re-used for both the sphere and a portion of the monkey. The monkey uses two layouts, one which has one UV map of a few faces, and another that has three maps.



How all the parts of UV Texturing work together

When using file textures, it is very important that you have Mapped the UVs of the mesh, and they are laid out appropriately.

You don't have to UV map the *entire* mesh. The sphere above on the left has some faces mapped, but other faces use procedural materials and textures. Only use UV Textures for those portions of your mesh where you want very graphic, precise detail. For example, a model of a vase only needs UV Texture for the rim where decorative artwork is incorporated. A throw pillow does not need a different image for the back as the front; in fact many throw pillows have a fabric (procedural material) back.

As another example, you should UV map both eyes of a head to the same image (unless you want one bloodshot and the other clear). Mapping both sides of a face to the same image might not be advisable, because the location of freckles and skin defects are not symmetrical. You could of course change the UV map for one side of the face to slightly offset, but it might be noticeable. Ears are another example where images or section of an images can be mapped to similar faces.

Workflow

The process consists of the following steps.

1. Create the Mesh. Unwrap it into one or more UV Layouts.
2. Create one or more Materials for the Mesh.
3. Create one or more images for each UV Layout and aspect of the texture. Either
   - paint directly on the mesh using Texture Paint in the 3D window,
   - load and/or edit an image in the UV Editor window, or
   - Bake the existing materials into an image for the UV Editor window.
4. Apply those images as UV Textures to the mesh to affect one or more aspects of the mesh. This is done by using one or more of the numerous Map To options. For example,
   - map to Color to affect the diffuse coloring of the mesh,
   - map to Nor to affect the normal direction to give the surface a bumpy or creased look, or
   - map to Spec (specularity) to make certain areas look shiny and oily.
5. Layer the Textures to create a convincing result.

Using Images and Materials

To use an image as the color and alpha (transparency) of the texture, you can create an image in an external paint program and tell the UV/Image Editor to Open that file as the texture, or you can create a New image and save it as the texture.

If you want to start off by creating an image using an external paint program, you will want to save an outline of your UV faces by using the Save UV Face Layout tool located in the UVs menu. This is discussed here.

Creating an Image Texture

To create an image within Blender, you have to first create a New Blank Image with a uniform color or test grid. After that, you can color the image using the:

- Vertex colors as the basis for an image
- Render Bake image based on how the mesh looks in the scene

After you have created your image, you can modify it using Blender's built-in Texture Paint or any external image painting program.

See Texture in 3D View but does not Render
You may be able to see the texture in Textured display mode in the 3D View; this is all that is required to have textures show up in Blender's Game Engine. Rendering, however, requires a material. You must have a Face Textures material assigned to the mesh for it to render using the UV Texture. In the Material settings, ADD NEW material to a selected object and enable Face Textures.

Examples

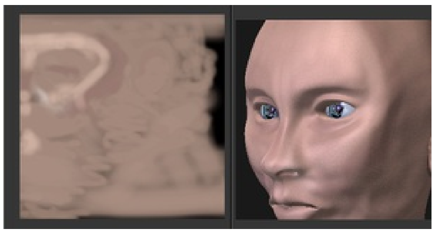
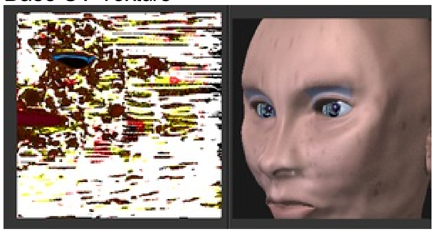
There may be one UV Layout for the face of a character, and another for their clothes. Now, to texture the clothes, you need to create an image at least for the Color of the clothes, and possible a "bump" texture to give the fabric the appearance of some weave by creating a different image for the Normal of the clothes. Where the fabric is worn, for example at the elbows and knees, the sheen, or Specularity, of the fabric will vary and you will want a different image that tells Blender how to vary the Specularity. Where the fabric is folded over or creased, you want another image that maps Displacement to the mesh to physically deform the mesh. Each of these are examples of applying an image as a texture to the mesh.

As another example, the face is the subject of many questions and tutorials. In general, you will want to create a Material that has the basic skin color, appropriate shaders, and sub-surface scattering. Then you will want to layer on additional UV Textures for:

- Freckle map for Color and Normal aspects
- Subdermal veins and tendons for Displacement
- Creases and Wrinkles and skin cell stratification for Normal
- Makeup images for Color
- Oily maps for Specularity
- For a zombie, Alpha transparency where the flesh has rotted away *(ewwww...)*
- Under chin and inside nostrils that receive less Ambient light
- Thin skin is more translucent, so a map is needed for that

Each image is mapped by using another Texture Channel. Each of these maps are images which are applied to the different aspects (Color, Normal, Specularity) of the image. Tileable images can be repeated to give a smaller, denser pattern by using the Texture controls for repeat or size.

Layering UV Textures


Base UV Texture

Layered UV Texture

Great textures are formed by layering images on top of one another. You start with a base layer, which is the base paint. Each successive layer on top of that is somewhat transparent to let the bottom layers show through, but opaque where you want to add on to details.

To avoid massive confusion, all image textures for a mesh usually use the same UV map. If you do, each image will line up with the one below it, and they will layer on top of one another like the examples shown to the right. To do this, just create one UV Texture (map) as described in this section. Then, create material image textures as described in the procedural materials section. Instead of mapping to Original Coordinates (OrCo), map to UV.

Use that map name repeatedly in the Material->Textures->Map Input panel by selecting UV and typing the name in the text field. In the example to the right, our UV Texture is called "Head" (you may have to expand the image to see the panel settings). Then, the image texture shown will be mapped using the UV coordinates. In the "Base UV Texture" example to the right, the face has two textures UV mapped; one for a base color, and another for spots, blemishes and makeup.

Both textures use the same UV Texture map as their Map Input, and both affect Color. The Makeup texture is transparent except where there is color, so that the base color texture shows through. Note that the colors were too strong on the image, so they amount of Col affects is turned down to 60% in the second layer (the blemish layer).

Normally, we think of image textures affecting the color of a mesh. Realism and photo-realistic rendering is a combination of many different ways that light interacts with the surface of the mesh. The image texture can be Mapped To not only color, but also Normal (bumpiness) or Reflection or any of the other attributes specified in the Map To panel.

If you paint a grey-scale image (laid out according to the UV Layout) with white where the skin is oily and shiny, and dark where it is not, you would map that input image according to the UV Layout, but have it affect Specularity (not color).

To make portions of a mesh transparent and thus reveal another mesh surface underneath, you would paint a grey-scale image with black where you want the texture transparent, map input to UV, and map it to Alpha (not color). To make portions of a mesh, like a piece of hot metal, appear to glow, you would use a grey-scale image mapped to Emit.

Believe it or not, this is only "the tip of the iceberg!" If everything that's been described here just isn't enough for you, the *texture nodes* feature, introduced in recent versions of Blender, enables you to layer and combine textures in almost any way you can imagine.

Mix and Match Materials

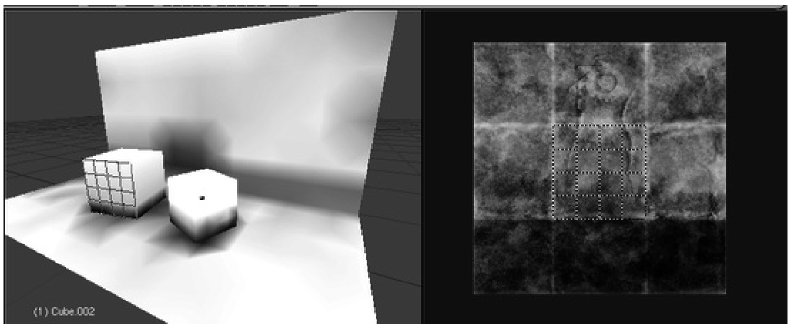
You can mix and match procedural materials and textures, vertex paint, and UV textures onto the same mesh.

The image to the right has a world with a red ambient light. The material has both VCol Paint and Face Textures enabled, and receives half of ambient light. A weak cloud texture affects color, mixing in a tan color. The right vertices are vertex painted yellow and the left is unpainted procedural gray. The UV Texture is a stock arrow image from the public domain texture CD. Scene lighting is a white light off to the right. From this information and the User Manual thus far, you should now be able to recreate this image.

You can also assign multiple materials to the mesh based on which faces you want to be procedural and which you want to be texture-mapped. Just don't UV map the faces you want to be procedural.

You can use UV Textures and VertexPaint (V in the 3D View window) simultaneously, if both are enabled in the Material settings. The vertex colors are used to modulate the brightness or color of the UV image texture:

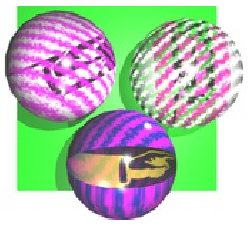
- UV Texture is at the base *(Face Textures)*
- Vertex paint affects its colors, then
- Procedural textures are laid on top of that,
- Area lights shine on the surface, casting shadows and what not, and finally
- Ambient light lights it up.



Vertex colors modulate texture.

A UV Layout can only have one image, although you can tile and animate the image. Since a layout is a bunch of arranged UV Maps, and a UV Map maps many mesh faces, a face can therefore only have one UV Texture image, and the UV coordinates for that face must fit entirely on the image. If you want a face to have multiple images, split the face into parts, and assign each part its own image. *(Or* you can get fancy with Nodes, but that's another story ...)

Using Alpha Transparency


Alpha UV Textures

Alpha 0.0 (transparent) areas of a UV Image render as black. Unlike a procedural texture, they do not make the base material transparent, since UV Textures do not operate on the base procedural material. The UV texture overrides any procedural color underneath. Procedural Textures are applied on top of UV Textures, so a procedural image texture would override any UV Texture. Transparent (black) areas of a procedural texture mapped to alpha operate on top of anything else, making the object transparent in those places. The only thing that modulates visible parts of a UV Texture are the Vertex Colors. In the example to the right, the finger image is transparent at the cuff and top of the finger and is used as a UV Texture. All three balls have a base material of blue and a marbling texture. The base material color is not used whenever Face Textures is enabled.

The top left ball has not had any vertex painting, and the finger is mapped to the middle band, and the texture is mapped to a pink color. As you can see, the base material has VCol Paint and Face Textures enabled; the base color blue is not used, but the texture is. With no vertex painting, there is nothing to modulate the UV Texture colors, so the finger shows as white. Transparent areas of the UV Image show as black.

The top right ball has had a pink vertex color applied to the vertical band of faces (in the 3D View window, select the faces in UV Paint mode, switch to Vertex Paint mode, pick a pink color, and Paint->Set Vertex Colors). The finger is mapped to the middle vertical band of faces, and VCol and Face Textures are enabled. The texture is mapped to Alpha black and multiplies the base material alpha value which is 1.0. Thus, white areas of the texture are 1.0, and 1.0 times 1.0 is 1.0 (last time I checked, at least), so that area is opaque and shows. Black areas of the procedural texture, 0.0, multiply the base material to be transparent. As you can see, the unmapped faces (left and right sides of the ball) show the vertex paint (none, which is gray) and the painted ones show pink, and the middle stripe that is both painted and mapped change the white UV Texture areas to pink. Where the procedural texture says to make the object transparent, the green background shows through. Transparent areas of the UV Texture insist on rendering black.

The bottom ball uses multiple materials. Most of the ball (all faces except the middle band) is a base material that does not have Face Textures (nor Vertex Color Paint - VCol Paint) enabled. Without it enabled, the base blue material color shows and the pink color texture is mixed on top. The middle band is assigned a new material (2 Mat 2) that *does* have vertex paint and Face Textures enabled. The middle band of faces were vertex painted yellow, so the white parts of the finger are yellow. Where the pink texture runs over the UV texture, the mixed color changes to green, since pink and yellow make a green.

If you want the two images to show through one another, and mix together, you need to use Alpha. The base material can have an image texture with an Alpha setting, allowing the underlying UV Texture to show through.

To overlay multiple UV images, you have several options:

- Create multiple UV Textures which map the same, and then use different images (with Alpha) and blender will overlay them automatically.
- Use the Composite Nodes to combine the two images via the AlphaOver node, creating and saving the composite image. Open that composited image as the UV Texture.
- Use an external paint program to alpha overlay the images and save the file, and load it as the face's UV Texture
- Define two objects, one just inside the other. The inner object would have the base image, and the outer image the overlaid image with a material alpha less than one (1.0).
- Use the Material nodes to combine the two images via the AlphaOver or Mix node, thus creating a third noded material that you use as the material for the face. Using this approach, you will not have to UV map; simply assign the material to the face using the Multiple Materials

UV Textures vs. Procedural Textures

A Material Texture, that has a Map Input of UV, and is an image texture that is mapped to Color, is equivalent to a UV Texture. It provides much more flexibility, because it can be sized and offset, and the degree to which it affects the color of your object can be controlled in the Map To panel. In addition, you can have different images for each texture channel; one for color, one for alpha, one for normals, one for specularity, one for reflectivity, *etc.* Procedural textures, like Clouds, are INCREDIBLY simple and useful for adding realism and details to an image.

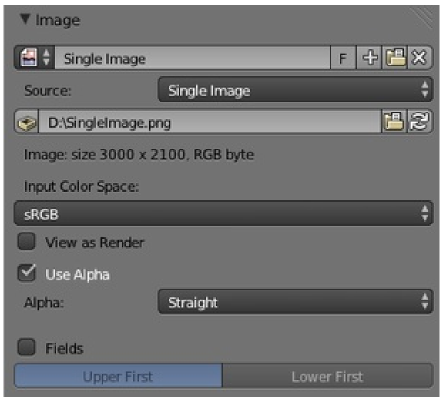
UV Texture	Procedural Texture
Image maps to precise coordinates on the selected faces of the mesh	Pattern is generated dynamically, and is mapped to the entire mesh (or portion covered by that material)
The Image maps once to a range of mesh faces specifically selected	Maps once to all the faces to which that material is assigned; either the whole mesh or a portion
Image is mapped once to faces.	Size XYZ in the MapInput allows tiling the texture many times across faces. Number of times depends on size of mesh
Affect the color and the alpha of the object.	Can also affect normals (bumpiness), reflectivity, emit, displacement, and a dozen other aspects of the mesh's appearance; can even warp or stencil subsequent textures.
Can have many for a mesh	Can be layered, up to 10 textures can be applied, layering on one another. Many mix methods for mixing multiple channels together.
Any Image type (still, video, rendered). Preset test grid available	Many different presents: clouds, wood grain, marble, noise, and even magic.
Provides the UV layout for animated	Noise is the only animated procedural texture

textures	Noise is the only animated procedural texture
Takes very limited graphics memory	Uses no or little memory; instead uses CPU compute power

So, in a sense, a single UV texture for a mesh is simpler but more limited than using multiple textures (mapped to UV coordinates), because they do one specific thing very well: adding image details to a range of faces of a mesh. They work together if the procedural texture maps to the UV coordinates specified in your layout. As discussed earlier, you can map multiple UV textures to different images using the UV Coordinate mapping system in the Map Input panel.

Settings

Image


▼ Image

Single Image F

Source: Single Image

D:\SingleImage.png

Image: size 3000 x 2100, RGB byte

Input Color Space:

sRGB

☐ View as Render

☑ Use Alpha

Alpha: Straight

☐ Fields

Upper First | Lower First

Image panel

In the Image Sampling panel we tell Blender which source file to use.

Image or Movie Datablock

Browse
: Select an image or video among linked to the .blend file

Name field
: Internal name of image

F
: Create a fake user for the image texture

+
: Replace active texture with a new one

Folder
: Browse for an image on your computer

X
: Unlink this image or movie.

Source
: Where the image come from. What kind of source file to use.

Generated
: Generated image in Blender.

Movie
: Movie file.

Image Sequence
: Multiple image files as a sequence.

Single Image
: Single image file.

File for Image or Movie texture
: See about supported Image formats.

Pack image
: Embed image into current .blend file

Path
: Path to file

File Browser
: Find a file on your computer. Hold ⇧ Shift to open the selected file and Ctrl to browse a containing directory.

Reload
: Reloads the file. Useful when an image has been rework in an external application.

Input Color Space
: Color space of the image or movie on disk

XYZ
: XYZ space.

VD16
: The simple video conversion from a gamma 2.2 sRGB space.

sRGB

Standart RGB display space.
Raw
Raw space.
Non-Color
Color space used for images which contains non-color data (i.e. normal maps).
Linear ACES
ACES linear space.
Linear
709 (full range). Blender native linear space.

View as Render
Apply render part of display transformation when displaying this image on the screen.

Use Alpha
Use the alpha channel information from the image or make image fully opaque

Straight
Transparent RGB and alpha pixels are unmodified.
Premultiplied
Transparent RGB pixels of an image are multiplied by the image's alpha value.

Fields
Work with field images. Video frames consist of two different images (fields) that are merged. This option ensures that when Fields are rendered, the correct field of the image is used in the correct field of the rendering. MIP Mapping cannot be combined with Fields.

Upper First
Order of video fields – upper field first.
Lower First
Order of video fields – lower field first.

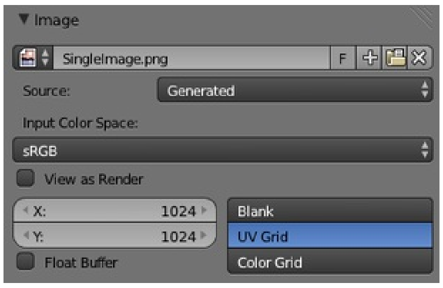


Image panel for Generated source of Image texture

For Generated source there are the specific options:

X and Y size
Width and height of image to be generated.
Generated Image Type
Which kind of image to be generated

Blank
Generate a blank image.
UV Grid
Generated grid to test UV mappings.
Color Grid
Generated improved UV grid to test UV mappings.

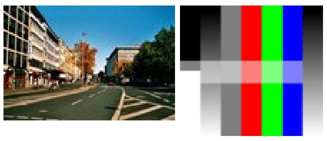
Float Buffer
Generate floating point buffer.

About specific options for **movie** and **image sequence** source see here

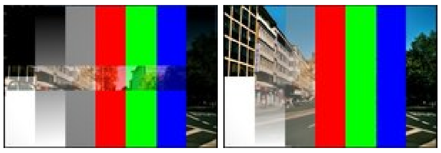
Image Sampling

In the Image Sampling panel we can control how the information is retrieved from the image.

The two images presented here are used to demonstrate the different image options. The *background image* is an ordinary JPG-file, the *foreground image* is a PNG-file with various alpha and greyscale values. The vertical bar on the right side of the foreground image is an Alpha blend, the horizontal bar has 50% alpha.


Left: Background image
Right: Foreground image

Alpha
: Options related to transparency

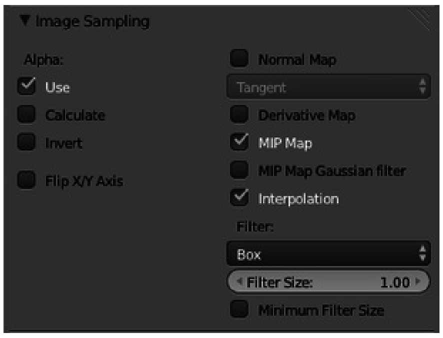

Left: Foreground image with Use alpha. The alpha values of the pixels are evaluated
Right: Foreground image with Calculate alpha

Use
: Works with PNG and TGA files since they can save transparency information (Foreground Image with UseAlpha). Where the alpha value in the image is less than 1.0, the object will be partially transparent and stuff behind it will show.

Calculate
: Calculate an alpha based on the RGB values of the Image. Black (0,0,0) is transparent, white (1,1,1) opaque. Enable this option if the image texture is a mask. Note that mask images can use shades of gray that translate to semi-transparency, like ghosts, flames, and smoke/fog.

Invert
: Reverses the alpha value. Use this option if the mask image has white where you want it transparent and vice-versa.

Flip X/Y Axis
: Rotates the image 90 degrees counterclockwise when rendered.


▼ Image Sampling

Alpha:
- ✓ Use
- Calculate
- Invert
- Flip X/Y Axis

- Normal Map
- Tangent
- Derivative Map
- ✓ MIP Map
- MIP Map Gaussian filter
- ✓ Interpolation
- Filter: Box
- Filter Size: 1.00
- Minimum Filter Size

Image Sampling panel

Normal Map
: This tells Blender that the image is to be used to create the illusion of a bumpy surface, with each of the three RGB channels controlling how to fake a shadow from a surface irregularity. Needs specially prepared input pictures. See Bump and Normal Maps.

Normal Map Space

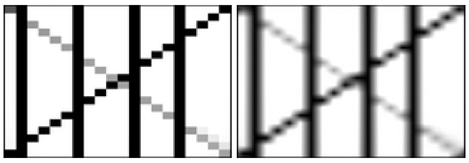
Tangent
Object
World
Camera

Derivative Map
: Use red and green as derivative values.

MIP Map
: MIP Maps are pre-calculated, smaller, filtered Textures for a certain size. A series of pictures is generated, each half the size of the former one. This optimizes the filtering process. By default, this option is enabled and speeds up rendering (especially useful in the game engine). When this option is OFF, you generally get a sharper image, but this can significantly increase calculation time if the filter dimension (see below) becomes large. Without MIP Maps you may get varying pictures from slightly different camera angles, when the Textures become very small. This would be noticeable in an animation.

MIP Map Gaussian filter
: Used in conjunction with MIP Map, it enables the MIP Map to be made smaller based on color similarities. In the game engine, you want your textures, especially your MIP Map textures, to be as small as possible to increase rendering speed and frame rate.

Interpolation
: This option interpolates the pixels of an image. This becomes visible when you enlarge the picture. By default, this option is on. Turn this option OFF to keep the individual pixels visible and if they are correctly anti-aliased. This last feature is useful for regular patterns, such as lines and tiles; they remain 'sharp' even when enlarged considerably. When you enlarge this 10x10 pixel Image , the difference with and without Interpolation is clearly visible. Turn this image off if you are using digital photos to preserve crispness.


Enlarged Image texture without and with Interpolation

Filter
: The filter size used in rendering, and also by the options MipMap and Interpolation. If you notice gray lines or outlines around the textured object, particularly where the image is transparent, turn this value down from 1.0 to 0.1 or so.

Texture Filter Type
: Texture filter to use for image sampling. Just like a *pixel* represents a *pic*ture *el*ement, a *texel* represents a *tex*ture *el*ement. When a texture (2D texture space) is mapped onto a 3D model (3D model space), different algorithms can be used to compute a value for each pixel based on samplings from several texels.

Box
: A fast and simple nearest-neighbor interpolation known as Monte Carlo integration

EWA
: **E**lliptical **W**eighted **A**verage — one of the most efficient direct convolution algorithms developed by Paul Heckbert and Ned Greene in the 1980s. For each texel, EWA samples, weights, and accumulates texels within an elliptical footprint and then divides the result by the sum of the weights.

  Eccentricity
  : Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower

FELINE
: FELINE (**F**ast **El**liptical **Line**s), uses several isotropic probes at several points along a line in texture space to produce an anisotropic filter to reduce aliasing artifacts without considerably increasing rendering time.

  Probes
  : Number of probes to use. An integer between 1 and 256.
    Further reading: McCormack, J; Farkas, KI; Perry, R; Jouppi, NP (1999) *Simple and Table Feline: Fast Elliptical Lines for Anisotropic Texture Mapping*, WRL

Area
: Area filter to use for image sampling

  Eccentricity
  : Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower

Filter Size
: The filter size used by MIP Map and Interpolation

Minimum Filter Size
: Use Filter Size as a minimal filter value in pixels

Image Mapping

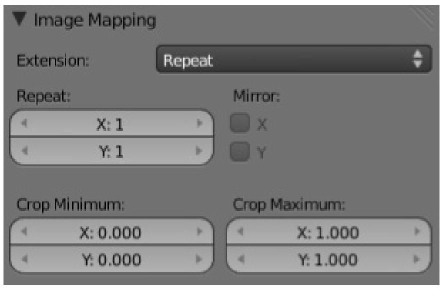


Image Mapping panel

In the Image Mapping panel, we can control how the image is mapped or projected onto the 3D model.

Extension

Extend
: Outside the image the colors of the edges are extended

Clip
: Clip to image size and set exterior pixels as transparent. Outside the image, an alpha value of 0.0 is returned. This allows you to 'paste' a small logo on a large object.

Clip Cube
: Clips to cubic-shaped area around the images and sets exterior pixels as transparent. The same as Clip, but now the 'Z' coordinate is calculated as well. An alpha value of 0.0 is returned outside a cube-shaped area around the image.

Repeat
: The image is repeated horizontally and vertically

  Repeat
  : X/Y repetition multiplier

  Mirror
  : Mirror on X/Y axes. This buttons allow you to map the texture as a mirror, or automatic flip of the image, in the corresponding X and/or Y direction.

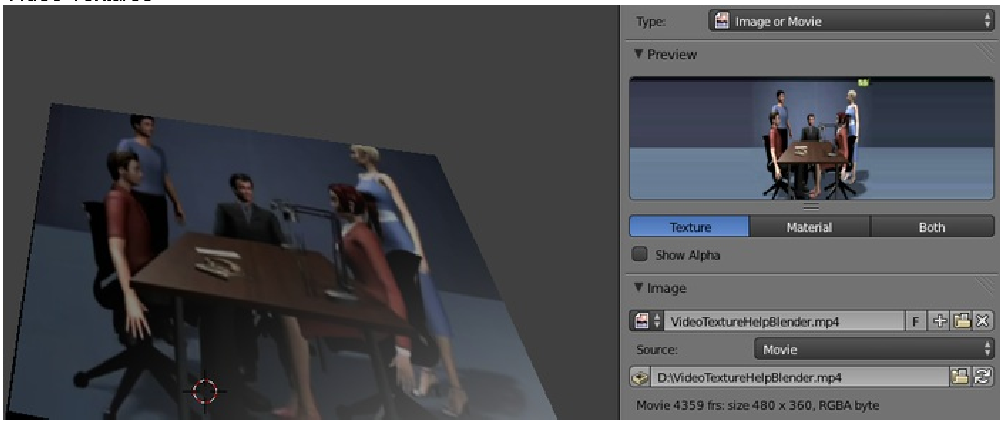
Checker
: Checkerboards quickly made. You can use the option size on the Mapping panel as well to create the desired number of checkers.

  Even/Odd
  : Set even/odd tiles

  Distance
  : Governs the distance between the checkers in parts of the texture size

Crop Minimum/Crop Maximum
: The offset and the size of the texture in relation to the texture space. Pixels outside this space are ignored. Use these to crop, or choose a portion of a larger image to use as the texture.

Video Textures


Type: Image or Movie
Preview
Texture
Material
Both
Show Alpha
Image
VideoTextureHelpBlender.mp4
Source: Movie
D:\VideoTextureHelpBlender.mp4
Movie 4359 frs: size 480 x 360, RGBA byte

Video texture

Video textures are a some kind of Image textures and based on movie file or sequence of successive numbered separate images. They are added in the same way that image textures are.

Options

Image

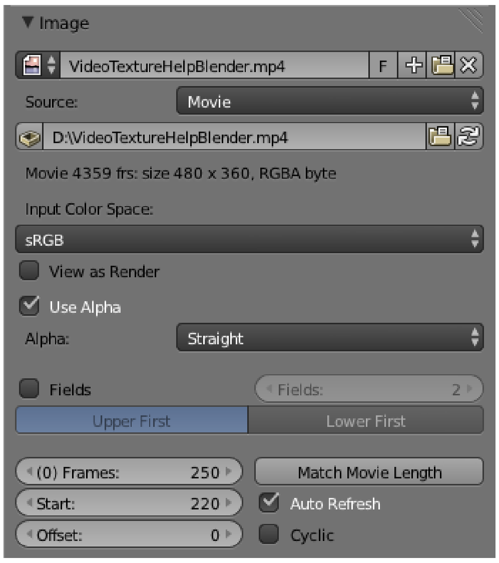


Image panel for video texture

Source
: For video texture the kind of source file to use is

  Movie
  : See about supported Movie formats.

  Image Sequence
  : See about supported Image formats.
    To load image sequence in any of the supported image file formats first click on the first frame and then Accept. Then change the Source to Image Sequence, and enter the ending frame number of this sequence.

More about loading source file for video texture see here.

Fields
: Work with field images. Video frames consist of two different images (fields) that are merged. This option ensures that when Fields are rendered, the correct field of the image is used in the correct field of the rendering.

  Upper First
  : Order of video fields – upper field first.

  Lower First
  : Order of video fields – lower field first.

  Fields
  : Number of fields per rendered frame. Used with Fields and interlaced video, it says whether each image has both odd and even, or just one.

Frames
: Number of frames/images in the movie or sequence to use

Start
: Global starting frame of the sequence/movie

Offset
: Offset the number of the frame to use in the animation. What frame number inside the movie/sequence to start grabbing.

Match Movie Length
: This button set image's user's length to the one of selected movie/sequence.

Auto Refresh
: Automatically refresh images on frame changes

Cyclic
: When the video ends, it will loop around the to the start and begin playing again.

For Movie source:

Use Alpha
: Use the alpha channel information from the image or make image fully opaque

  Straight
  : Transparent RGB and alpha pixels are unmodified.

  Premultiplied
  : Transparent RGB pixels of an image are multiplied by the image's alpha value.

About input color space for video texture see here.

About video sampling for video texture see here.

About video mapping for video texture see here.

Texture Nodes

As an alternative to using the Texture Stack, Blender includes a node-based texture generation system which enables you to create textures by combining colors, patterns and other textures in much the same way that you combine Material Nodes.

You can use these textures wherever you can use regular textures: you can place them in texture channels, in material nodes, in particle systems, and even inside other textures.

Nodes Concepts

Nodes

"Nodes" are individual blocks that perform a certain operation, and might have one or many different outputs.

Conceptually, there are three basic types of nodes:

- **Input Nodes**

  these nodes *produce* information, but do not have any inputs of their own.
  Examples are: *Render Layers*, *Value* and *RGB* nodes.

- **Processing Nodes**:

  these nodes *filter* or *transform* their inputs, to produce one or more outputs.
  Examples are: *RGB Curves*, *Defocus,'* and **Vector Blur** nodes.

- **Output Nodes**:

  these nodes *consume* their inputs to produce some kind of meaningful result.
  Examples are: *Composite* node (which determines the final output used by Blender), *Viewer* (which displays the output of a socket), and **File Output** node.

Noodles

The essential idea of nodes is that you can create an arbitrarily-complex *network* of nodes, by connecting the *outputs* of one or more nodes to the *inputs* of one or more other nodes. Then, you can set appropriate parameters (as you see fit) for each node.

This network is called a "noodle" and it describes how information literally *flows through* to produce whatever result you want.

Node Groups

You can define *node groups*, and use those groups as they were a single node.

You can link and append these node groups from other files.

Note
Node-based textures do **not** work for realtime display, they will only be visible in rendered images.

Using Texture Nodes

To use texture nodes with the current texture, open a Node Editor window, set it to Texture mode by clicking the “Texture” icon (▨) in its header.

To start adding nodes, you first need to select a material. Now you can either click the New button in the Node editor, or the New button in the texture panel. Once you have a texture selected, you can toggle it to function as a regular texture or a node texture by clicking the Use Nodes option in the Node Editor.

The default node setup will appear: a red-and-white checkerboard node connected to an Output named “`Default`”. For *texture* nodes, you can create as many Outputs as you like in your node setup. (Other types of node networks, as you may recall, are limited to only one Output node.) See the next section for details.

For instructions on how to add, remove and manipulate the nodes in the tree, see the Node Editor manual.

Using Multiple Outputs

Each texture that you define with Texture Nodes can have several outputs, which you can then use for different things. For example, you might want your texture to define both a diffuse (color) map and a normal map. To do this, you would:

1. Create two texture slots in the texture list, and set them to the same texture datablock.
2. Add two Output nodes to the node tree, and type new names into their Name text-boxes: *e.g.* “`Diffuse`” for one and “`Normal`” for the other.

3. Underneath the texture picker in the texture panel, you'll see a dropdown list with the names of your outputs. For each entry in the texture list, select the desired output by changing the menu entry (e.g. set on to "`Diffuse`" and the other to "`Normal`").

You can also use these named outputs if you've decided to define your material using Material Nodes. In this case, you probably won't be using Texture Channels. Instead, you'll insert Texture nodes into your Material Node tree using Add → Input → Texture. Then, inside the texture node that you've just added, you can select which output you want to use *(e.g.* `Diffuse` or `Normal`).

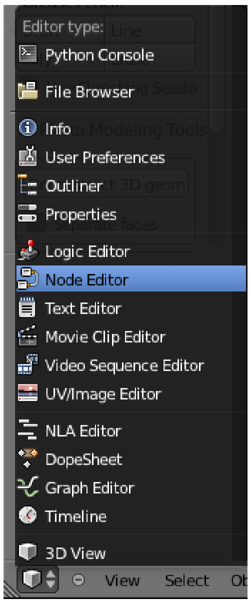
See also

- Development page

The Node Editor

This section explains the window in general, and its header menu options. It also tells you how to enable nodes for use within Blender.

Accessing The Node Editor

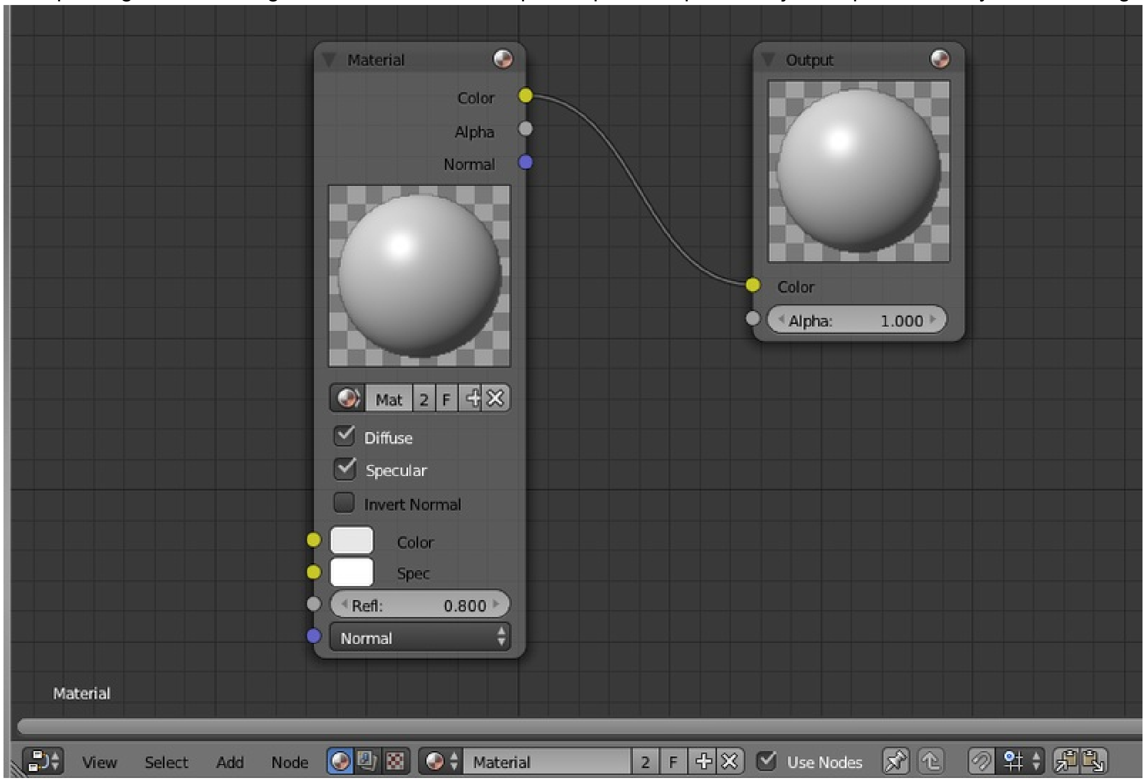


Select the Node Editor window.

First let's enter the node editor by changing our window type to Node Editor. As shown in *Select the Node Editor window*, click on the window type icon and select Node Editor from the popup list. Node maps can get quite large, so use or create a big window. The window has a graph-paper style background and a header.

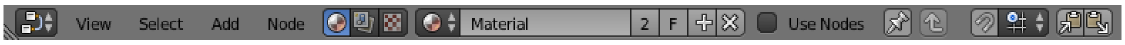
Each scene within your blend file can have multiple Material Node maps and ONE Compositing Node map. The Node Editor window shows either type of map, depending on the selector position.

Hint
You might want to add a new window layout called 6-Nodes (the list is shown on the User Preferences header at the top of your screen) comprised mostly of one big Node Editor window. My layout has the buttons window at the bottom and a text editor window on the side for me to keep notes. If you have a widescreen display (or even a regular one), you might also want to add a 3D view or UV/Image Editor window to the left side of the Node window layout, so you can work with images or your model while you're manipulating nodes. Having the 3D Preview Render panel open on top of an object is quite useful if you're tweaking material nodes.



Node Editor.

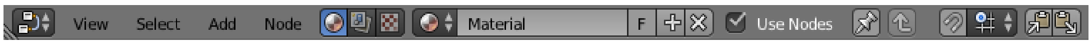
By default, the header, when first displayed, is uninitialized as shown:

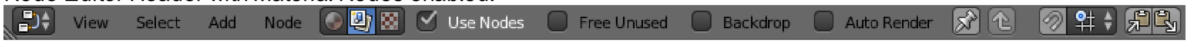


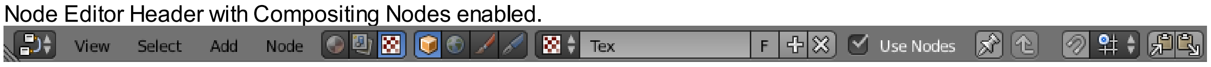
Default Node Editor header.

Activating Nodes

- What nodes to use?
  - If you want to work with a material node map, click the ball in the Material/Compositing node set selector. (See *Node Editor Header with Material Nodes enabled.*)
  - If you want to work with a compositing node map, click the overlaped pictures on the Material/Compositing node set selector. (See *Node Editor Header with Compositing Nodes enabled.*)
  - If you want to work with a texture node map, click the checker on the Material/Compositing node set selector. (See *Node Editor Header with Texture Nodes enabled.*)
- To actually activate nodes, click the Use Nodes button.
- The first time that you select either a Material, Compositing or a Texture node map, the Node Editor window will be instantly filled with starter input and output compositing nodes already connected together.



Node Editor Header with Material Nodes enabled.


Node Editor Header with Compositing Nodes enabled.


Node Editor Header with Texture Nodes enabled.

Node Editor Window Actions

When the cursor is in the window, several standard Blender hotkeys and mouse actions are available, including:

Popup menu
: Space - Brings up a main popup menu, allowing you to add, view, select, etc.

Delete
: X or Del - Deletes the selected node(s).

Box select
: B - Starts the bounding box selection process. Position your cursor and LMB click & drag to select a set of nodes.

Cut connections (lasso)
: CtrlAlt LMB click & drag - Starts a lasso selection, BUT when you let up the mouse button, all threads (connections) within the lasso are broken.

Undo
: CtrlZ Very helpful if you forgot to press B before box-selecting, eh?

Redo
: CtrlY or Ctrl⇧ ShiftZ - You can use this if you used "undo" a bit too often :)

Select multiple
: ⇧ Shift LMB or ⇧ Shift RMB - Multiple node select.

Grab/Move
: G - Moves your current selection around.

Standard Window Control
Node maps can get pretty hairy (large and complicated, that is). The contents of the window (the node map) can be panned just like any other Blender window by clicking MMB and dragging about. Wheeling Wheel up/down or using the keypad + NumPad/- NumPad will zoom in/out. The window can be resized and combined using the standard window techniques (see *Navigating in 3d Space*).

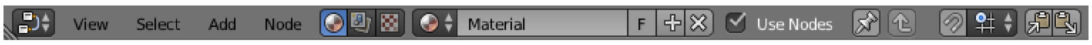
Node Editor Header

At a glance

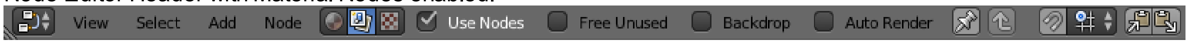
On the window header, you will see header options:

- View - to see things more clearly;
- Select - to do things more clearly;
- Add - to walk with...err..to add Nodes, organized by type;
- Node - to do things with selected nodes, akin to vertices;
- a Material, Compositing or Texture node set selector;
- a Use Nodes button;
- a Use Pinned button;
- a Go to Parent button;
- a Snap button;
- a Snap Node Element selector;

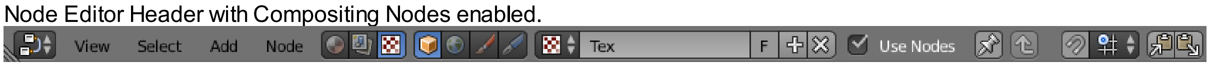
- a Copy Nodes button;
- a Paste Nodes button.



Node Editor Header with Material Nodes enabled.



Node Editor Header with Compositing Nodes enabled.



Node Editor Header with Texture Nodes enabled.

Menus

View, Select and Add

These popup menus provide the basic functions:

View
: This menu changes your view of the window, standing in for the standard keyboard shortcuts + NumPad (zoom in), - NumPad (zoom out), ↖ Home (zoom all) or equivalent mouse actions.

Select
: This menu allows you to select a node or groups of nodes, and does the same as typing the hotkey to select all A or start the border select B process.

Add
: This menu allows you to add nodes. Please see the next section for a discussion on the types of nodes that you can add, and what they do. Clicking this menu item is the same as pressing Space when the cursor is in the window

Node

Hide
: H - Hides your selected nodes. Just like vertices in a mesh.

Grouping
: Most importantly, this menu option allows you to create a user-defined group of nodes. This group can then be edited and added to the map. To create a group, select the nodes you want, and then Node → Make Group, or just use the keyboard shortcut CtrlG. Edit the name using the little input box in the group. Groups are easily identified by their green header and cool names you have picked for them.

Delete
: X - Deletes selected nodes.

Duplicate
: ⇧ ShiftD - Makes an Unlinked copy, with the same settings as the original.

Grab
: G - Moves the little nodes around according to your mouse, just like with meshes.

Duplicate - Faked you out
The new copy is placed **exactly over the old one**. But it isn't the connected one, so playing with the controls will do nothing to your images, even though it **looks** like it's connected with the little threads coming out of the node that is **underneath**. You have to move the duplicated node to reveal the connected node beneath it.

Grab - Reminder Only
Just like my mother-in-law, the menu item does not actually do anything; it's just there to remind you that you can press the G key when your cursor is in the window and actually accomplish something with your life (like rearranging nodes in the window).

Buttons

Material/Composite/Texture Selector

Nodes are grouped into two categories, based on what they operate on:

- to work with Material Nodes, click on the ball,
- to work with Compositing nodes, click on the overlaped pictures,
- to work with Texture nodes, click on the checker.

Use Nodes Button

This button tells the render engine to use the node map in computing the material color or rendering the final image, or not. If not, the map is ignored and the basic render of the material tabs or scene is accomplished.

Use Pinned Button

This button tells the render engine to use pinned node tree.

Go to Parent Button

This button allows you go to parent node tree.

Snap Button

Toggle snap mode for node in the Node Editor window.

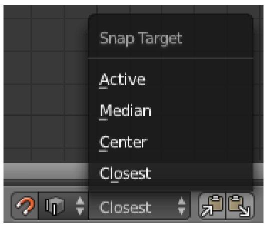
Snap Node Element selector
: This selector provide the follow node elements for snap:

  Grid (default)
  : Snap to grid of the Node Editor window.

  Node X
  : Snap to left/right node border.

  Node Y
  : Snap to top/bottom node border.

  Node X/Y
  : Snap to any node border.
  : Snapping to node border takes into account snap target:



Snap Target.

Snap Target
: Which part to snap onto the target
: Closest: Snap closest point onto target.
: Center: Snap center onto target.
: Median: Snap median onto target.
: Active: Snap active onto target.

Copy Nodes Button

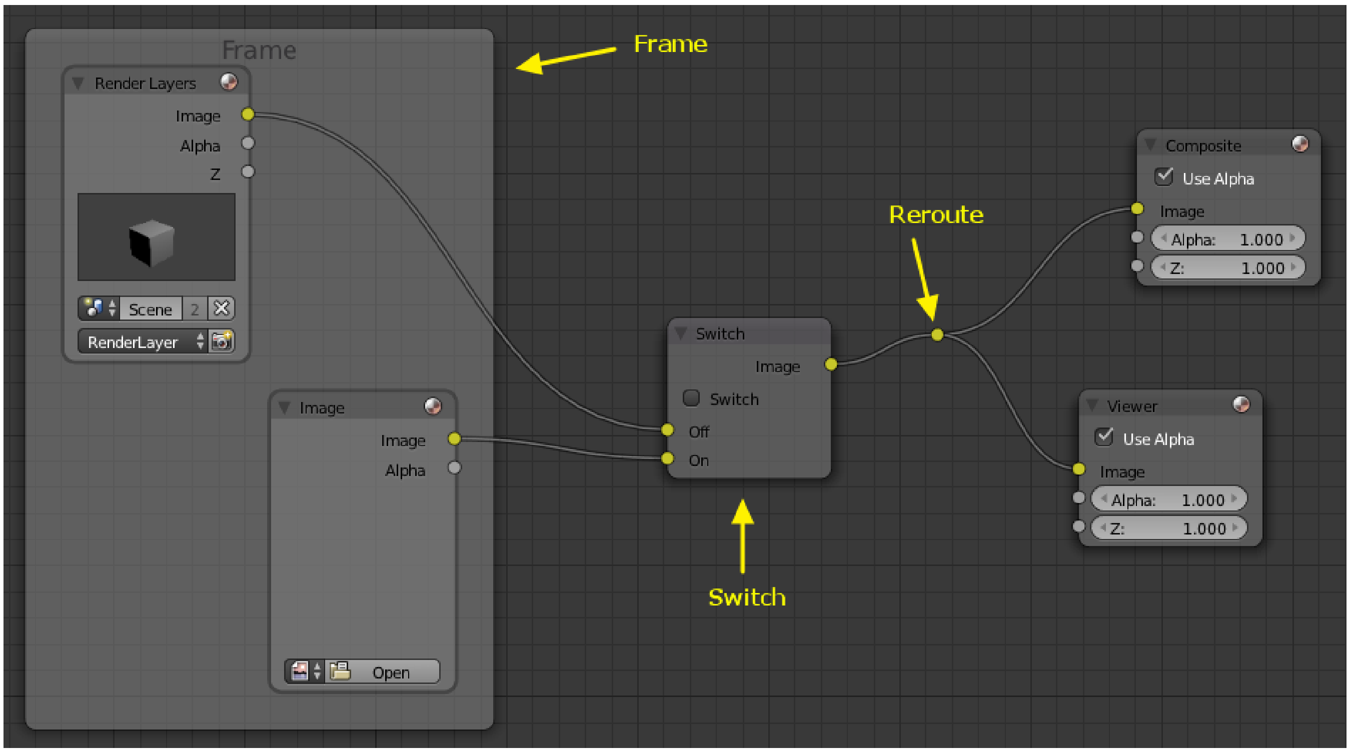
This button allows you copy selected nodes to the clipboard.

Paste Nodes Button

This button allows you paste nodes from the clipboard to the active node tree.

Layout Nodes

Layout nodes are designed for enhanced arrangement your nodes in Node Editor window. They are available from menu Add -> Layout.



Examples of the Layout Nodes.

Blender provides the following layout nodes:

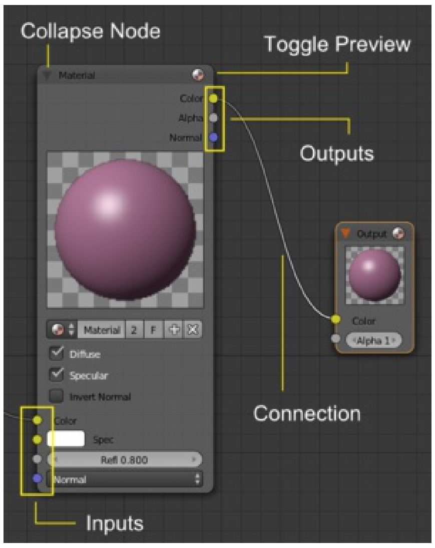
Frame
: This node is used for simple join a several nodes in the single place.

Reroute
: It's intended for branching threads with the purpose of optimization of Node Editor window's space.

Switch
: This node is applied for switching between color values. Available only for Compositing nodes.

Node Controls

This page explains the widgets to control a node.



Nodes main controls

Titlebar
This contains the node's name, along with several different collapse buttons.

Input sockets
The left side of a node has input sockets:

- *blue sockets* accept vectors.
- *yellowsockets* accept colors.
- *gray sockets* accept single values (like alpha).

Output sockets
The right side of a node has output sockets:

- *blue sockets* produce vectors.
- *yellowsockets* produce colors.
- *gray sockets* produce single values (like alpha).

Image preview
Inside the node there's an area to show the image preview being output by the node or the curves that control the node behavior (for example in a RGB node).

Buttons and menus
Below the image preview there are buttons and menus to control the node behavior.

Threads
A curved line shows a connection from an output socket to an input socket. The socket types must match.

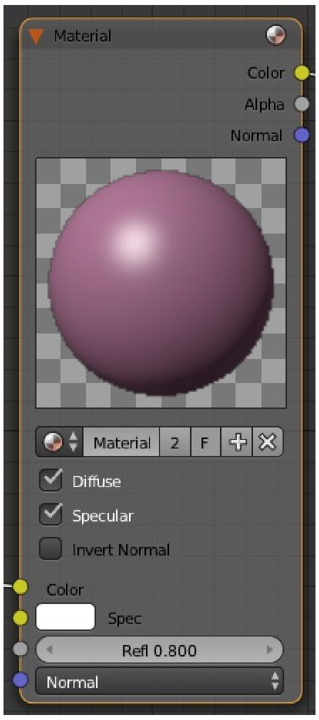
Connections associated with the active node are highlighted for better visibility.

Collapsing toggles

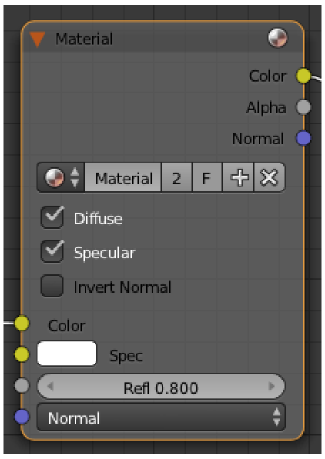
At the top of a node there are up to 4 visual controls for the node (*Top of a Node*). Clicking these controls influences how much information the node shows.

Node toggle (▼ ▶)
The arrow on the left collapses/uncollapses the node.

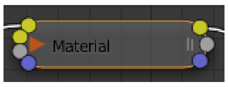
Preview image toggle
The sphere button on the far right of the titlebar hides/unhides the preview image.



Full display



Preview hidden

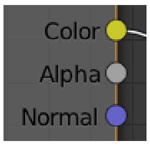


Node collapsed

Sizing the node

Fine Sizing of an individual node can also be accomplished somewhat by clicking LMB and dragging on the left or right edge of the node.

Sockets



Node Sockets.

Each Node in your node window will have "sockets" (often also referred to as "connectors") which are small colored circles to which input data and output data can be linked (*Node Sockets*).

The sockets on the left side of a node describe *inputs,* while the sockets on the right side are *outputs.*

For your convenience, nodes are *color-coded* according to the type of information they expect to send or receive. There are three colors:

● Yellow sockets
: Indicates that **color** information needs to be input or will be output from the node.

● Gray sockets
: Indicates values (**numeric**) information. It can either be a single numerical value or a so-called "value map." (You can think of a value map as a grayscale-map where the different amount of bright/dark reflects the value for each point.) If a single value is used as an input for a "value map" socket, all points of the map are set to this same value.
Common use: Alpha maps and value options for a node.

● Blue/Purple sockets
: Indicates **vector/coordinate/normal** information.

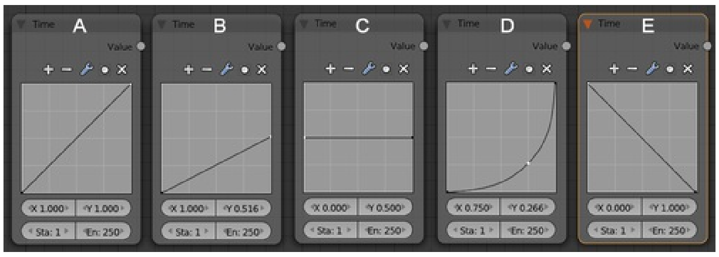
Between nodes, yellow must be linked to yellow, gray to gray, blue to blue, unless you use a *converter,* which we'll cover later on.

Next to the color in the node you will see the name of that socket. Though not always the case, you can think of the name of the socket as what the information is *intended* to be. But this is not necessarily what it *has* to be. For example, I can add a link from a gray socket titled Alpha to the material node's gray Reflection socket and still get a result, the key thing being that it's a "gray to gray" connection.

There are exceptions where you can mix yellow (i.e. a color image) and gray (*e.g.* grayscale) without converters. Blender normally places a converter if needed, so feel free to experiment with them. You can use the "Viewer" output nodes, as explained in the later sections, to see if/how it works.

Curves

Some nodes have a curve area that translates an input value to an output value. You can modify this curve shape by clicking on a control point and moving it, or adding a control point. Some examples are shown below:



Modifying a curve node.

Every curve starts out as a straight line with a slope of 1. The curve starts out with two tiny black control points at each end of the line. Clicking LMB on a control point selects it and it turns white.

Changing the curve affects how the output is generated. The input, X, usually proceeds linearly (at regular intervals) across the **bottom** axis. Go up until you hit the curve, and then over to the **right** to determine the Y output for that corresponding X. So, for the second example, as X goes from 0 to 1.0 across the bottom, Y varies from 0.0 to 0.5. In the third, as X goes from 0.0 to 1.0 across the bottom, Y stays constant at 0.5. So, in the picture above, these curves have the following effect on time: **A** don't affect, **B** slow down, **C** stop, **D** accelerate, and **E** reverse time.

The "Curves" widget is a built-in feature in Blender's UI, and can be used anywhere, provided the curve data itself is being delivered to this widget. Currently it is used in the Node Editor and in the UV Window.

This widget will map an input value horizontally and return the new value as indicated by the height of the curve.

Note: The fact that one of the points on the curve is "white" in each of these screenshots is *not* significant; it just means that it happened to be the point most-recently selected by your author when preparing this tutorial. What matters here is the shape of *the curve,* not the position (nor the color) of the control points that were used to define it.

RGB Curves

Multiple curves can be edited in a single widget. The typical use, RGB curves, has "Combined" result or "Color" ("C") as the first curve, and provides curves for the individual R, G, and B components. All four curves are active together; the "C" curve gets evaluated first.

Selecting curve points

- LMB always selects 1 point and deselects the rest.
- Hold ⇧ Shift while clicking to extend the selection or select fewer points.

Editing curves

- LMB click&drag on a point will move points.
- A LMB click on a curve will add a new point.
- Dragging a point exactly on top of another will merge them.
- Holding ⇧ Shift while dragging snaps to grid units.
- Ctrl LMB adds a point.
- Use the X icon to remove selected points.

Editing the view

The default view is locked to a 0.0-1.0 area. If clipping is set, which is the default, you cannot zoom out or drag the view. Disable clipping with the icon resembling a #.

- LMB click&drag outside of curve moves the view
- Use the + and - icons to zoom in or out.

Special tools

The wrench icon gives a menu with choices to reset a view, to define interpolation of points, or to reset the curve.

Using nodes

Adding Nodes

Nodes are added in two ways to the node editor window:

- By clicking the Add menu in the node editor toolbar and picking the type of node you want, or
- By clicking the ⇧ ShiftA -> Add and picking a node from the popup Add menu.

Arranging Nodes

In general, try to arrange your nodes within the window such that the image flows from left to right, top to bottom. Move a node by clicking on a benign area and dragging it around. The node can be clicked almost anywhere and dragged about; connections will reshape as a bezier curve as best as possible.

Connecting nodes

LMB -click and drag a socket: you will see a branch coming out of it: this is called a "thread".

Kepp dragging and connect the thread to an input socket of another node, then release the LMB .

In this case, a copy of each output is routed along a thread. However, only a single thread can be linked to an input socket.

Disconnecting nodes

To break a link between sockets Ctrl LMB -click in an empty areas near the thread you want to disconnect and drag: you will see a little cutter icon appearing at your mouse pointer. Move it over the thread itself, and release the LMB .

Duplicating a node

Click LMB or RMB on the desidered node, press ⇧ ShiftD and move the mouse away to see the duplicate of the selected node appeing under the mouse pointer.

Gotcha!
When you duplicate a node, the new node will be positioned *exactly* on top of the node that was duplicated. If you leave it there (and it's quite easy to do so), you can **not** easily tell that there are *two* nodes there! When in doubt, grab a node and move it slightly to see if something's lurking underneath.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in the node editor, making your material or composite 'noodle' (node network) easier to work with. Grouping nodes also creates what are called NodeGroups (inside a .blend file) or NodeTrees (when appending).

Conceptually, "grouping" allows you to specify a *set* of nodes that you can treat as though it were "just one node." You can then re-use it one or more times in this or some other .blend file(s).

As an example: If you have created a material using nodes that you would like to use in another .blend file, you *could* simply append the material from one .blend file to another. However, what if you would like to create a new material, and use a branch from an existing material node network? You could re-create the branch. Or you could append the material to the new .blend file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different .blend files. What if you have created a “Depth of Field” composite node network and would like to use it in another .blend file? What if you wanted to apply exactly the same series of operations dozens of times? Here again, you *could* re-create the network, but this is not very efficient. A better method of re-use, for either material node branches or composite node networks, would be to create groups of nodes.

Once a group has been defined, it becomes an opaque object; a reusable software component. You can (if you choose) ignore exactly how it is *defined,* and simply use it (as many times as you like) for whatever it *does.* Groups can be made available through the Blender library and standard appending method.

Grouping Nodes

Panel: Node Editor

Menu: ⇧ ShiftA → Group → Make Group

To create a node group, in the node editor, select the nodes you want to include, then press CtrlG or ⇧ ShiftA » Group » Make Group. A node group will have a green title bar. All of the selected nodes will now be minimized and contained within the group node. Default naming for the node group is *NodeGroup, NodeGroup.001* etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one .blend file to another, Blender does not make a distinction between material node groups or composite node groups, so I recommend some naming convention that will allow you to easily distinguish between the two types. For example, name your material node branches *Mat_XXX,* and your composite node networks *Cmp_XXX.*

What not to include in your groups (all types of Node editors)

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not include**:

Source nodes
if you include a source node in your group, you'll end up having the source node appearing *twice:* once inside the group, and once outside the group in the new material node-network.

Examples of source nodes are: the *Material Node* (Material nodes editor) and the *Render Layers Node* (Composite Editor).

Output node
if you include an output node in the group, there won't be an output socket available *from* the group!

Examples of output nodes are: the *Output Node* (Material nodes editor) and the *Viewer Node* (Composite Editor).

Editing Node Groups

With a group node selected, pressing ⇆ Tab expands the node to a window frame, and the individual nodes within it are shown to you. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of your editor window. You will not be able to thread them to an outside node directly from them; you have to use the external sockets on the side of the Group node. To add or remove nodes from the group, you need to ungroup them.

Ungrouping Nodes

The AltG command destroys the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

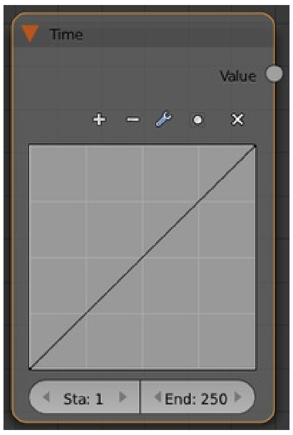
Appending Node Groups

Once you have appended a NodeTree to your .blend file, you can make use of it in the node editor by pressing ⇧ ShiftA → Add → Group, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Input Nodes

Input nodes provide input data for other nodes.

Time



Time node

The time node uses a frame range to output a value between 0 and 1. By default the node output a linear transition from 0 to 1 from frame 1 to 250. The shape of the curve can be manipulated to vary the output over time in different ways.

+
Zoom in.

-
Zoom out

Tools

Reset View

Resets curve view

Vector Handle

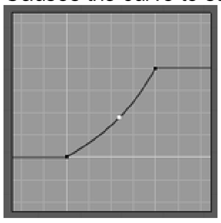
Breaks tangent at curve handle, making a angle.

Auto Handle

Default smooth interpolation of curve segments

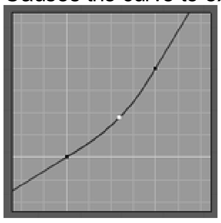
Extend Horizontal

Causes the curve to stay horizontal before the first point and after the last point.


Extend Horizontal

Extend Extrapolated

Causes the curve to extrapolate before the first point and after the last point, based on the shape of the curve.


Extend Extrapolate

Reset Curve

Resets shape of curve to original linear shape.

Clipping Options

Use Clipping

Forces curve points to stay between specified values.

Min X/Y and Max X/Y

Set the minimum and maximum bounds of the curve points.

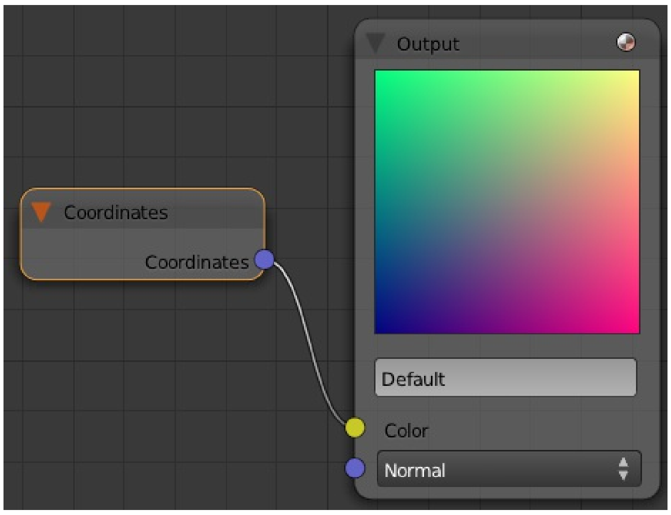
X
Delete curve points. The first and last points cannot be deleted.

X and Y
The coordinates of the selected edit point.

Sta
Specify the start frame to use.

End
Specify the end frame to use.

Coordinates

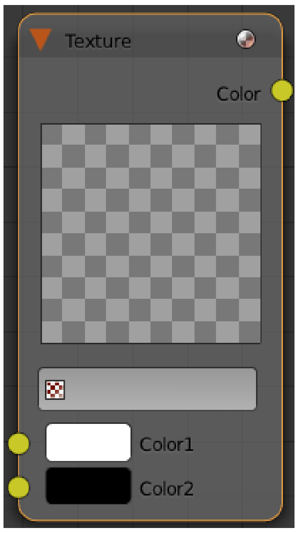


Coordinates node

The Coordinates node outputs the geometry local coordinates, relative to its bounding box as RGB colors:

- Red channel corresponds to X value.
- Green channel corresponds to Y value.
- Green channel corresponds to Z value.

Texture Node



Texture node

The texture node can be used to load a another node based or non-node based texture.

Color 1 and Color 2
These can be used to remap a greyscale texture using two colors.

Image Node

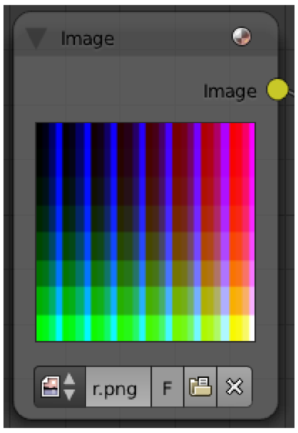


Image node

The image node can be used to load an external image.

Browse for image
: Select an image that already exists in the scene.

Datablock name
: Set the name of the image datablock.

F
: Save this image datablock, even if it has no users.

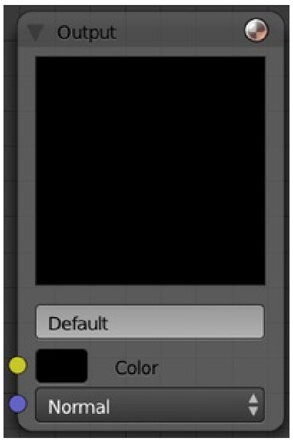
Open image
: Select image to use from file browser.

Unlink datablock
: Remove the image datablock from the node.

Texture Output Nodes

These node serves a outputs for node textures

Output



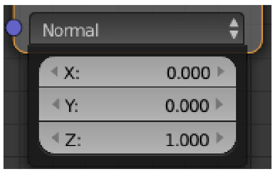
Output node

This node contains the result of the node texture. Multiple output nodes can exist in a node texture, however only one of them is active. The active one is set in the Texture Panel in the Output drop down.

Controls

Name node box
: The name box for naming this texture node. See also Using Multiple Outputs for Texture Node.

Color selector button
: Opens the default color selector for choosing color if any didn't select to output moment.



Normal control of Output node

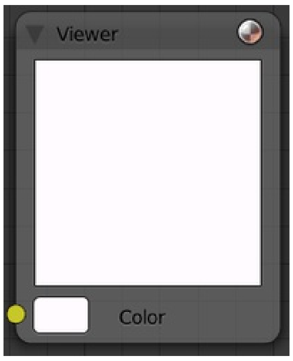
Normal
: 3D-direction of the face in relation to the camera. The value can be provided by another node or set manually.

Inputs

Color
: The color data that the texture renders

Normal
: The normal map that the texture will output.

Viewer



Viewer node

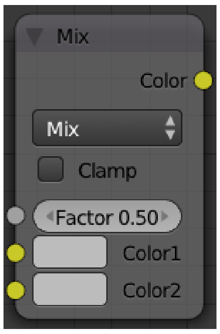
The viewer node can be used to preview the results of a node.

Inputs

Color
: Preview the color data of the results of a node. If nothing input this, that avaible *Color* selector button for choosing color here.

Texture Color Nodes

Mix



mix node

This node mixes a base color or image (threaded to the top socket) together with a second color or image (bottom socket) by working on the individual and corresponding pixels in the two images or surfaces. The way the output image is produced is selected in the drop-down menu. The size (output resolution) of the image produced by the mix node is the size of the base image. The alpha and Z channels (for compositing nodes) are mixed as well.

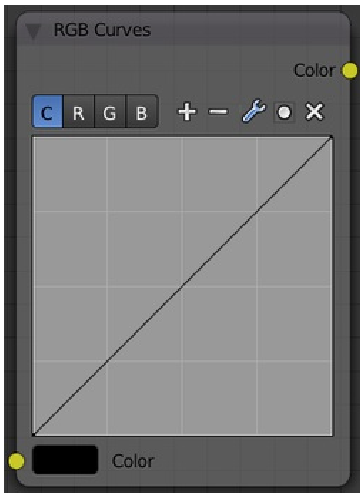
Mix	The background pixel is covered by the foreground using alpha values.
Add	The pixels are added together. Fac controls how much of the second socket to add in. Gives a bright result. The "opposite" to Subtract mode.
Subtract	The foreground pixel (bottom socket) is subtracted from the background one. Gives a dark result. The "opposite" to Add mode.
Multiply	Returns a darker result than either pixel in most cases (except one of them equals white=1.0). Completely white layers do not change the background at all. Completely black layers give a black result. The "opposite" to Screen mode.
Screen	Both pixel values are inverted, multiplied by each other, the result is inverted again. This returns a brighter result than both input pixels in most cases (except one of them equals 0.0). Completely black layers do not change the background at all (and vice versa) - completely white layers give a white result. The "opposite" of Multiply mode.
Overlay	A combination of Screen and Multiply mode, depending on the base color.
Divide	The background pixel (top socket) is divided by the second one: if this one is white (= 1.0), the first one isn't changed; the darker the second one, the brighter is the result (division by 0.5 - median gray - is same as multiplication by 2.0); if the second is black (= 0.0, zero-division is impossible!), Blender doesn't modify the background pixel.
Difference	Both pixels are subtracted from one another, the absolute value is taken. So the result shows the distance between both parameters, black stands for equal colors, white for opposite colors (one is black, the other white). The result looks a bit strange in many cases. This mode can be used to invert parts of the base image, and to compare two images (results in black if they are equal).
Darken	Both pixels are compared to each other, the smaller one is taken. Completely white layers do not change the background at all, and completely black layers give a black result.
Lighten	Both parameters are compared to each other, the larger one is taken. Completely black layers do not change the image at all and white layers give a white result.
Dodge	Some kind of inverted Multiply mode (the multiplication is replaced by a division of the "inverse"). Results in lighter areas of the image.
Burn	Some kind of inverted Screen mode (the multiplication is replaced by a division of the "inverse"). Results in darker images, since the image is burned onto the paper, er..image (showing my age).
Color	Adds a color to a pixel, tinting the overall whole with the color. Use this to increase the tint of an image.
Value	The RGB values of both pixels are converted to HSV values. The values of both pixels are blended, and the hue and saturation of the base image is combined with the blended value and converted back to RGB.
Saturation	The RGB values of both pixels are converted to HSV values. The saturation of both pixels are blended, and the hue and value of the base image is combined with the blended saturation and converted back to RGB.
Hue	The RGB values of both pixels are converted to HSV values. The hue of both pixels are blended, and the value and saturation of the base image is combined with the blended hue and converted back to RGB.

Color Channels
There are two ways to express the channels that are combined to result in a color: RGB or HSV. RGB stands for the Red,Green,Blue pixel format, and HSV stands for Hue,Saturation,Value pixel format.

Clamp
: Clamps the result of the mix operation between 0 and 1. Some of the mix types can produce reults above 1 even if the inputs are both between 0 and 1, such as Add.

Factor
: The amount of mixing of the bottom socket is selected by the Factor input field (Fac:). A factor of zero does not use the bottom socket, whereas a value of 1.0 makes full use. In Mix mode, 50:50 (0.50) is an even mix between the two, but in Add mode, 0.50 means that only half of the second socket's influence will be applied.

RGB Curves

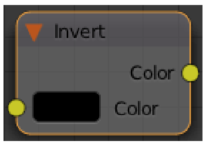


RGB Curves node

For each color component channel (RGB) or the composite (C), this node allows you to define a bezier curve that varies the input (across the bottom, or x-axis) to produce an output value (the y-axis). By default, it is a straight line with a constant slope, so that .5 along the x-axis results in a .5 y-axis output. Click and drag along the curve to create a control point and to change the curve's shape. Use the X to delete the selected (white) point.

Clicking on each C R G B component displays the curve for that channel. For example, making the composite curve flatter (by clicking and dragging the left-hand point of the curve up) means that a little amount of color will result in a lot more color (a higher Y value). Effectively, this bolsters the faint details while reducing overall contrast. You can also set a curve just for the red, and for example, set the curve so that a little red does not show at all, but a lot of red does.

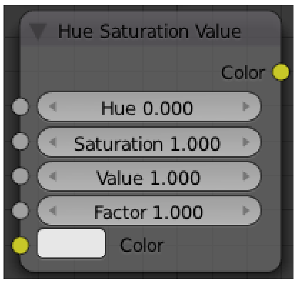
Invert



invert node

This node simply inverts the input values and colors.

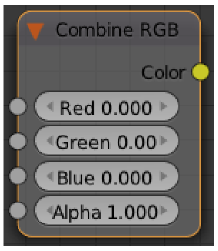
Hue Saturation Value



Hue Saturation Value node

Use this node to adjust the Hue, Saturation, and Value of an input.

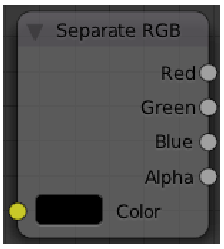
Combine and Separate RGB



Combine RGB node

These two nodes allow you to convert between float values and color values. Colors are composed of 3 or 4 channels; red, green, blue, and sometimes alpha.

With Combine RGB, you can specify the values of each channel, and the node will combine them into a color value.

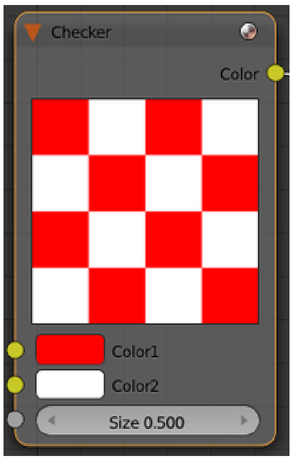


Separate RGB node

With Separate RGB, you can specify a color value, and get each channel value out of it.

Pattern Nodes

Checker



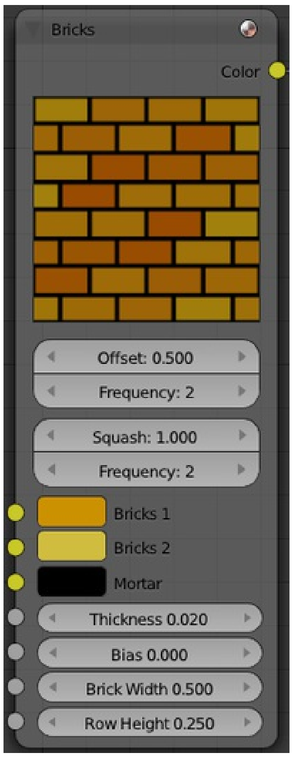
Checker node

The checker node creates a checkerboard pattern

color 1/color 2
Sets the color of the squares

Size
The scale of the checker pattern

Bricks



Bricks node

The Bricks node creates a brick like pattern

Offset
The relative offset of the next row of bricks

Frequency
Offset every N rows. The brick pattern offset repeats every N rows.

Squash
Scales the bricks in every N rows by this amount.

Frequency
Squash every N rows.

Bricks 1, Bricks 2

Sets the color range of the bricks. Brick colors are chosen randomly between these two colors.

Mortar
: Sets the mortar color, in between the bricks.

Thickness
: Sets the thickness of the mortar

Bias
: The bias of randomly chosen colors, between -1 and 1. -1 Makes all bricks Color 1, and a value of 1 makes them all Color 2.

Brick Width
: Sets the horizontal size of all the bricks.

Row Height
: Sets the verticalsize of all the bricks.

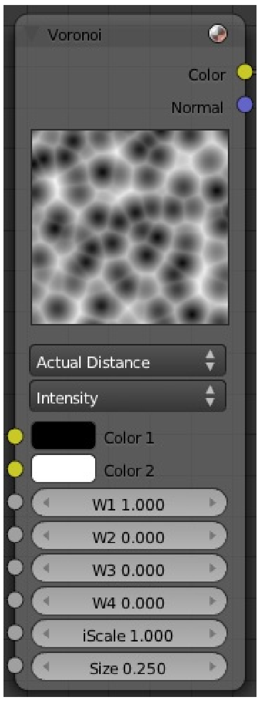
Texture Nodes

These nodes generat procedural textures, and function just like their non node based counterparts.

Common Options

Color 1/Color 2
: Remaps the procedural texture with these colors. These do not function in the Magic node.

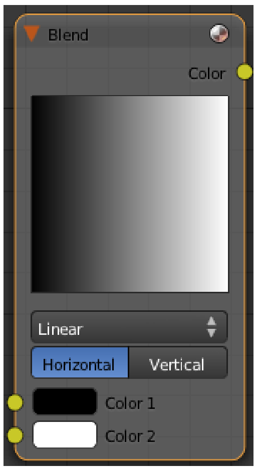
Voronoi



Voronoi node

See Here

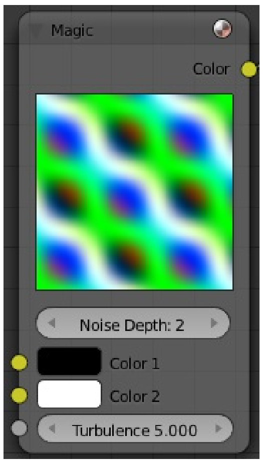
Blend



Blend node

See Here

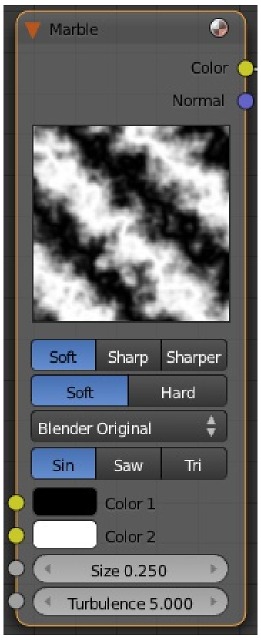
Magic



Magic node

See Here

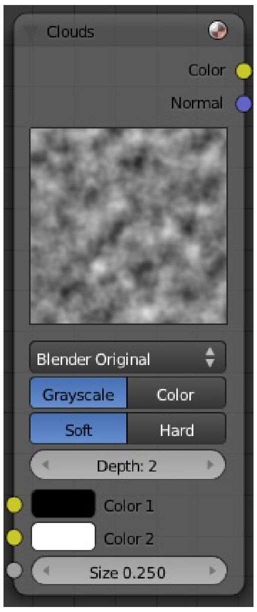
Marble



Marble node

See Here

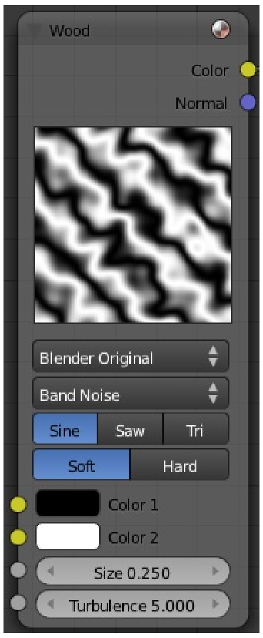
Clouds



Clouds node

See Here

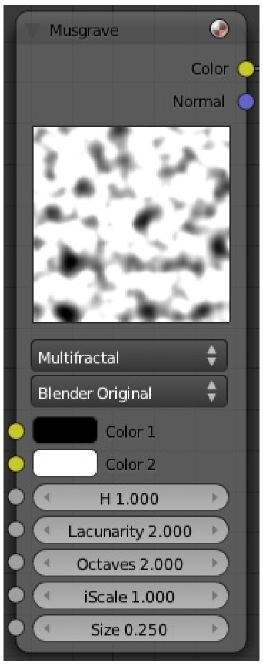
Wood



Wood node

See Here

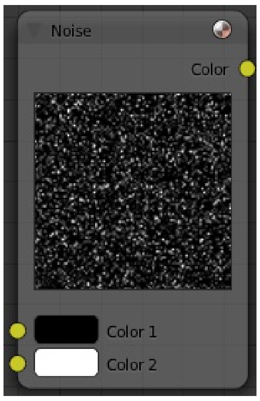
Musgrave



Musgrave

See Here

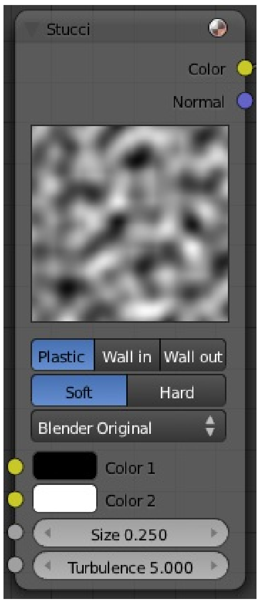
Noise



Noise

See Here

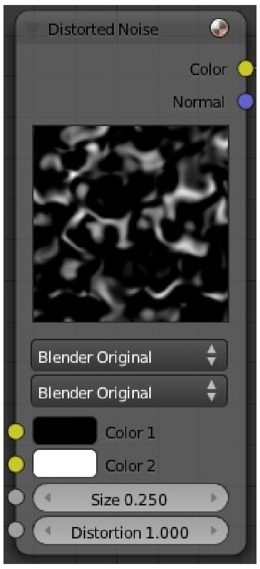
Stucci



Stucci

See Here

Distorted Noise



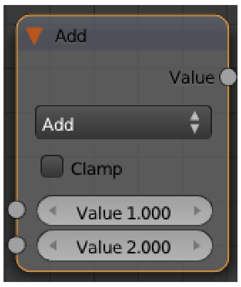
Distorted Noise node

See Here

Texture Convertor Nodes

As the name implies, these nodes convert the colors in the material in some way.

Math



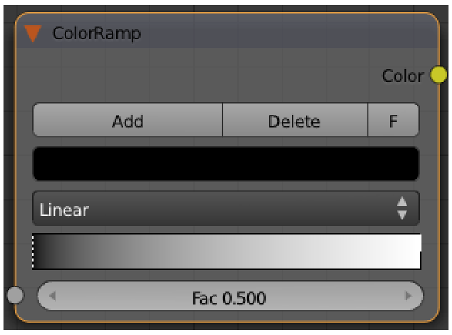
math node

The math node performs one of several math functions on one or two inputs

Clamp
    Clamps the result between 0 and 1.

Add
    Add the two inputs
Subtract
    Subtract input 2 from input 1
Multiply
    Multiply the two inputs
Divide
    Divide input 1 by input 2
Sine
    The sine of input 1 (degrees)
Cosine
    The cosine of input 1 (degrees)
Tangent
    The tangent of input 1 (degrees)
Arcsine
    The arcsine (inverse sine) of input 1 (degrees)
Arccosine
    The arccosine (inverse cosine) of input 1 (degrees)
Arctangent
    The arctangent (inverse tangent) of input 1 (degrees)
Power
    Input 1 to the power of input 2 (input1^input2)
Logarithm
    log base input 2 of input 1
Minimum
    The minimum of input 1 and input 2
Maximum
    The maximum of input 1 and input 2
Round
    Rounds input 1 to the nearest integer
Less Than
    Test if input 1 is less than input 2, returns 1 for true, 0 for false
Greater Than
    Test if input 1 is greater than input 2, returns 1 for true, 0 for false

ColorRamp Node



ColorRamp Node

The ColorRamp Node is used for mapping values to colors with the use of a gradient. It works exactly the same way as a Colorband for textures and materials, using the Factor value as a slider or index to the color ramp shown, and outputting a color value and an alpha value from the output sockets.

By default, the ColorRamp is added to the node map with two colors at opposite ends of the spectrum. A completely black black is on the left (Black as shown in the swatch with an Alpha value of 1.00) and a whitewash white is on the right. To select a color, LMB click on the thin vertical line/band within the colorband. The example picture shows the black color selected, as it is highlighted white. The settings for the color are shown above the colorband as (left to right): color swatch, Alpha setting, and interpolation type.

To change the hue of the selected color in the colorband, LMB click on the swatch, and use the popup color picker control to select a new color. Press ↵ Enter to set that color.

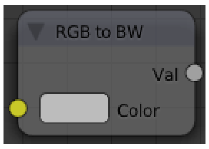
To add colors, hold Ctrl down and LMB click inside the gradient. Edit colors by clicking on the rectangular color swatch, which pops up a color-editing dialog. Drag the gray slider to edit Alpha values. Note that you can use textures for masks (or to simulate the old "Emit" functionality) by connecting the alpha output to the factor input of an RGB mixer.

To delete a color from the colorband, select it and press the Delete button.

When using multiple colors, you can control how they transition from one to another through an interpolation mixer. Use the interpolation buttons to control how the colors should band together: Ease, Cardinal, Linear, or Spline.

Use the A: button to define the Alpha value of the selected color for each color in the range.

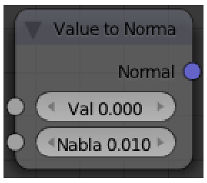
RGB to BW Node



RGB to BW Node

This node converts a color image to black-and-white by computing the luminance of the rgb values.

Value to Normal



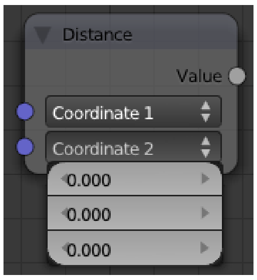
Value to Normal node

Computes a normal map based on greyscale values of an input

Val
: The texture to compute the normal map from

Nabla
: Size of derivative offset used for calculating normals.

Distance



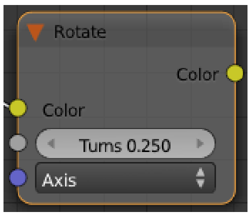
Distance node. Coordinate 2 dropdown is displayed

Computes the distance between two 3d coordinates.

Distort Nodes

These nodes allow you to change the mapping of a texture.

Rotate



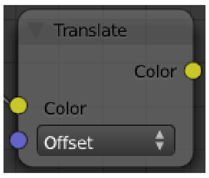
Rotate node

Rotate the texture coordinates of an image or texture.

Turns
: The number of times to rotate the coordinates 360 degrees about the specified axis.

Axis
: The axis to rotate the mapping about

Translate

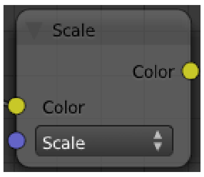


Translate node

Translate the texture coordinates of an image or texture.

Offset
: The amount to offset the coordinates in each of the 3 axes.

Scale

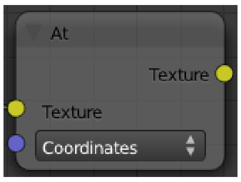


Scale node

Scale the texture coordinates of an image or texture.

Scale
: The amount to scale the coordinates in each of the 3 axes.

At



Returns the color of a texture at the specified coordinates. If the coordinates are not spatially varying, the node will return a single color.

Coordinates
: The point at which to sample the color. For images, the space is between -1 and 1 for x and y.

Volume Textures

Blender has two textures that can be applied to volumetric data:

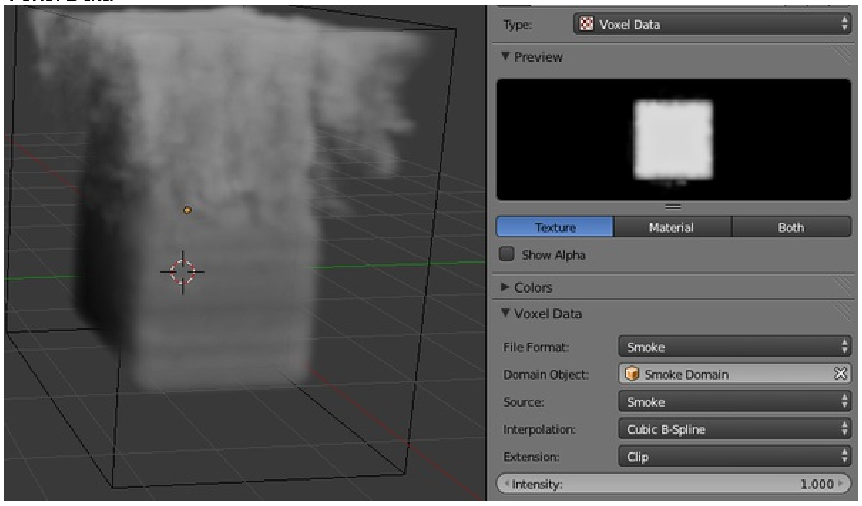
Voxel Data

Voxel data renders a voxel source. It can be used for rendering Blender's internal smoke simulations. Other sources include binary raw formats, and Image Sequence, which can be used to stack a sequence of images into a 3D representation

Point Density

Point density renders a given point cloud (object vertices or particle system) as a 3D volume

Voxel Data



Type: Voxel Data
File Format: Smoke
Domain Object: Smoke Domain
Source: Smoke
Interpolation: Cubic B-Spline
Extension: Clip
Intensity: 1.000

Voxel Data texture

Voxel data renders a voxel source, working very similarly to an image texture, but in 3d. Various input data source types are available (such as smoke voxel data, or external files), as well as various interpolation methods.

The voxels are stored in a flat z/y/x grid of floats. Functions for sampling this based on location within the (0,1) bounds are available in:

- source/blender/blenlib/intern/voxel.c

The default voxel data source, Smoke, is used for rendering Blender's internal smoke simulations. Other sources include binary raw formats, and Image Sequence, which can be used to stack a sequence of images into a 3D representation, which is a common format for medical volume data such as CT scans.

Settings

File Format
: Blender Voxel

  Default binary voxel file format.

  8 bit RAW

  8 bit grayscale binary data.

  Image Sequence

  Generate voxels from a sequence of image slices.

  Smoke

  Render voxels from a Blender smoke simulation.

Source Path
: The external source data file to use for 8 bit Raw data and Blender Voxel formats

Domain Object (Smoke)
: Object used as the smoke simulation domain

Source
: Smoke

  Use smoke density and color as texture data.

  Flame

  Use flame temperature as texture data.

  Heat

  Use smoke heat as texture data. Values from -2.0 to 2.0 are used.

  Velocity

  Use smoke velocity as texture data.

Resolution
: Resolution of the voxel grid when using 8 bit Raw data.

Interpolation
: Nearest Neighbor

No interpolation, fast but blocky and low quality.

Linear

Good smoothness and speed.

Quadratic

Mid-range quality and speed.

Cubic Catmull-Rom

Smoothed high quality interpolation, but slower.

Extension

Extend

Extend by repeating edge pixels of the image.

Clip

Clip to image size and set exterior pixels as transparent.

Repeat

Cause the image to repeat horizontally and vertically.

Intensity

Multiplier for intensity values

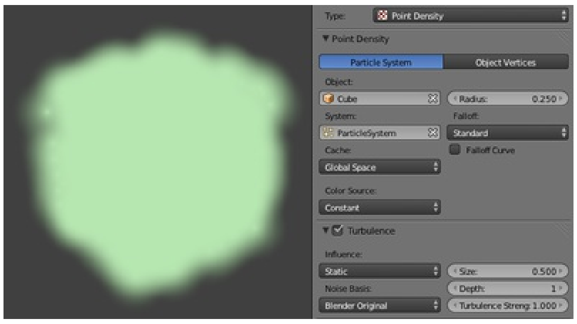
Point Density Texture

Point density renders a given point cloud (object vertices or particle system) as a 3D volume, using a user-defined radius for the points. Internally, the system uses a BVH data structure for fast range lookups.

The rendered points are spherical by default, with various smooth falloff options, as well as simple Turbulence options for displacing the result with noise, adding fine detail. When using Point Density with a particle system, additional particle info such as particle velocity, age, and speed, can be visualized using a color/alpha ramp gradient.

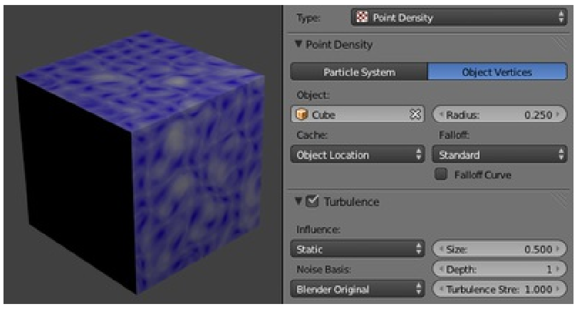
Options

Point Density


Type: Point Density
Point Density
Particle System | Object Vertices
Object: Cube
Radius: 0.250
System: ParticleSystem
Falloff: Standard
Cache: Global Space
Falloff Curve [ ]
Color Source: Constant
Turbulence [✓]
Influence: Static
Size: 0.500
Noise Basis: Blender Original
Depth: 1
Turbulence Streng: 1.000

Point Density texture with Particle System

Particle System
: Particle System, Generate point density from a particle system.


Type: Point Density
Point Density
Particle System | Object Vertices
Object: Cube
Radius: 0.250
Cache: Object Location
Falloff: Standard
Falloff Curve [ ]
Turbulence [✓]
Influence: Static
Size: 0.500
Noise Basis: Blender Original
Depth: 1
Turbulence Stre: 1.000

Point Density texture with cloud Object Vertices

Object Vertices
: Object Vertices, Generate point density from an object’s vertices.

Object
Radius
System
Falloff

- Standard
- Smooth
- Soft
  - Softness
- Constant
  - Density is constant within lookup radius.
- Root
- Particle Age
- Particle Velocity
  - Velocity Scale

Falloff Curve
: Use a custom falloff

Cache
: Coordinate system to cache particles in
: Global Space
: Emit Object Space

Emit Object Location

Color Source
: Data to derive the color results from
: Constant

Constant color

Particle Age

Lifetime mapped as 0.0 - 1.0 intensity.

Particle Speed

Particle speed (absolute magnitude of velocity) mapped as 0.0-1.0 intensity.
Scale

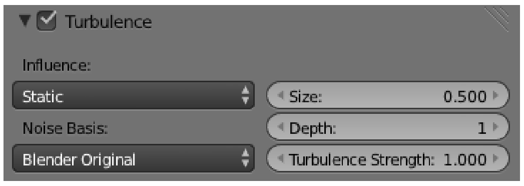
Multiplier to bring particle speed within an acceptable range.

Particle Velocity

XYZ velocity mapped to RGB colors.
Scale

Multiplier to bring particle speed within an acceptable range.

Turbulence



Turbulence panel

Adds directed noise to the density at render time

Influence
: Method for driving added turbulent noise
: Static

Noise patterns will remain unchanged, faster and suitable for stills.

Particle Velocity

Turbulent noise driven by particle velocity.

Particle Age

Turbulent noise driven by the particle’s age between birth and death.

Global Time

Turbulent noise driven by the global current frame.

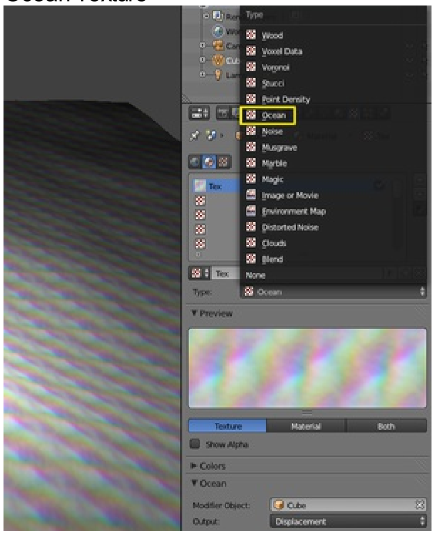
Noise Basis
: See Here

Size
: Scale of the turbulent noise

Depth
: Level of detail in the added turbulent noise

Turbulence Strength
: Strength of the added turbulent noise

Ocean Texture

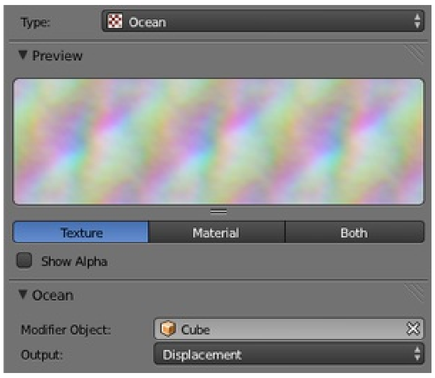


Ocean texture

Texture generated by an Ocean modifier.

Options

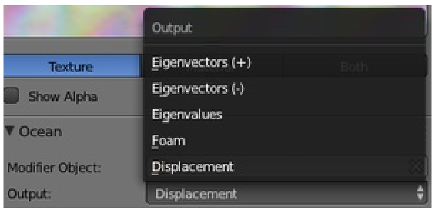
Ocean panel



Ocean texture Options

Modifier Object
: Object containing the ocean modifier.

Output
: The data that is output by the texture



Ocean texture Output settings

Displacement

Output XYZ displacement in RGB channels

Foam

Output foam (wave overlap) amount in single channel

Eigenvalues

Positive Eigenvalues

Eigenvector (-)

Negative Eigenvectors

Eigenvector (+)

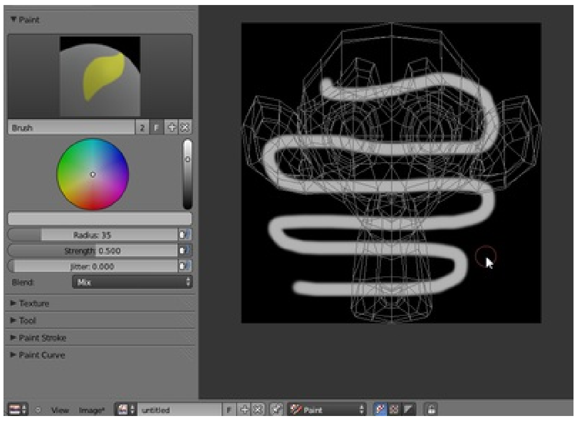
Positive Eigenvectors

Texture Painting

A UV Texture is a picture (image, sequence or movie) that is used to color the surface of a mesh. The UV Texture is mapped to the mesh through one or more UV maps. There are three ways to establish the image used by the UV Texture:

- Paint a flat image in the UV/Image Editor onto the currently selected UV Texture, using its UV map to transfer the colors to the faces of the mesh.
- Paint the mesh in the 3D View, and let Blender use the currently selected UV map to update the UV Texture (see "Projection Painting").
- Use any image-editing (paint) program to create an image. In the UV/Image Editor, select the UV Texture and load the image. Blender will then use that texture's UV map to transfer the colors to the faces of the mesh

Blender features a built-in paint mode called Texture Paint which is designed specifically to help you edit your UV Textures and images quickly and easily in either the UV/Image Editor window or the 3D View window. Since a UV Texture is just a special-purpose image, you can also use any external paint program. For example, GIMP is a full-featured image manipulation program that is also open-source.



Texture painting in Blender

Since a mesh can have layers of UV Textures, there may be many images that color the mesh. However, each UV Texture only has one image.

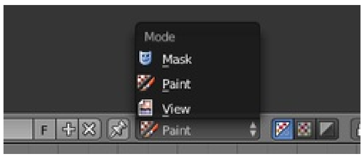
Texture Paint works in both a 3D window and the UV/Image Editor window. In the 3D window in Texture Paint mode, you paint directly on the mesh by projecting onto the UVs.

Getting Started

Once you have unwrapped your model to a UV Map (as explained in previous pages), you can begin the texturing process. You cannot paint on a mesh in Texture Paint mode without **first** unwrapping your mesh, **and** doing one of the following steps. Either:

- Load an image into the UV/Image Editor (Image->Open->select file), or
- Create a new image (Image->New->specify size).

After you have done one of these two things, you can modify the image using the Texture Paint mode:



Enabling paint mode

- In the 3D View window, select Texture Paint mode from the mode selector in the window header, and you can paint directly onto the mesh.
- In the UV/Image Editor window, switch the editing context from View to Paint (shown to the right).

Square Power of 2
Texture paint is very fast and responsive when working in the 3D window and when your image is sized as a square where the side lengths are a power of two, e.g. 256x256, 512x512, 1024x1024, etc.

Once you enable Texture Painting, your mouse becomes a brush. To work with the UV layout (for example, to move coordinates) you must go back to "View" mode.

As soon as you enable Texture Painting or switch to Texture Paint mode, brush settings become available in the Toolbar Panel (T-key).

In the UV/Image Editor window, you paint on a flat canvas that is wrapped around the mesh using UV coordinates. Any changes made in the UV/Image Editor window show up immediately in the 3D window, and vice versa.

A full complement of brushes and colors can be selected from the Properties panel in the UV/Image Editor. Brush changes made in either panel are immediately reflected in the other panel. However, the modified texture will **not** be saved automatically; you must explicitly do so by Image->Save in the UV/Image Editor window.

Texture Preview

If your texture is already used to color, bump map, displace, alpha-transparent, etc., a surface of a model in your scene (in other techie words, is mapped to some aspect of a texture via a texture channel using UV as a map input), you can see the effects of your painting in the context of your scene as you paint.

To do this, set up side-by-side windows, one window in 3D View set to Textured display mode, and the second UV/Image Editor window loaded with your image. Position the 3D View to show the object that is UV mapped to the loaded image. Open a Preview window (see 3D View Options for more info) and position it over the object. In the image to the right, the texture being painted is mapped to the "Normal" attribute, and is called "bump mapping", where the gray-scale image is used to make the flat surface appear bumpy. See Texture Mapping Output for more information on bump mapping.

Brushes Settings

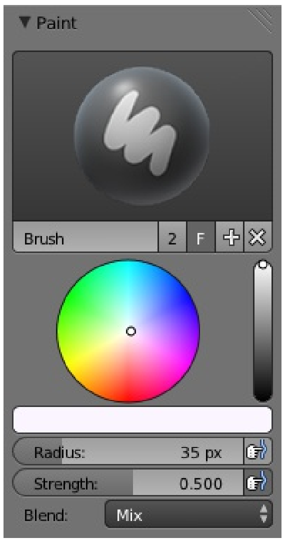
Press T in the UV/Image Editor to show the Toolbar panel. With this panel, you can create many brushes, each with unique settings (such as color and width). Use the Brush selector to switch between brushes, or to create a new brush. When you add a brush, the new brush is a clone of the current one. You can then change the setting for the new brush. Texture paint has an unlimited number of brushes and unique user-defined controls for those brushes which can be set in the Paint Tool panel.

To use a brush, click on its name. Use the selector up/down arrow, if there are more brushes on the flyout window than can be displayed at once. Name your brush by clicking on the name field and entering any name you wish, such as "Red Air" for a red airbrush. To toss out a brush, click the brush delete X button next to its name. If you want to keep this brush around for the next time you run Blender, click the Fake user button next to the brush delete X button.

If you have a tablet pen with pressure sensitivity, toggle the small "P" button next to the opacity, size, falloff and spacing buttons to control these parameters using the pressure of the pen. Using your pen's eraser end will toggle on the Erase Alpha mode.

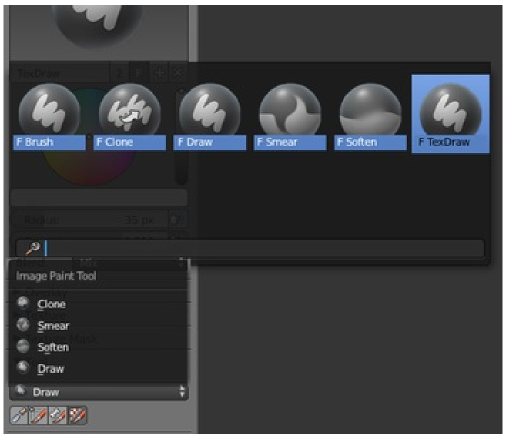
Press S on any part of the image to sample that color and set it as the brush color.

Brush



Brush Settings

Brush presets
: Select a preset brush. Most brushes have common settings.



Types of brushes

Types of brushes

There are four different types of brushes

Draw
: the normal brush; paints a swath of color

Soften
: blends edges between two colors

Smear
: when you click, takes the colors under the cursor, and blends them in the direction you move the mouse. Similar to the "smudge" tool of *Gimp*.

Clone
: copies the colors from the image specified (Tex.Dirt in the example), to the active image. The background image is shown when this brush is selected; use the Blend slider to control how prominent the background image is.

Enable Pressure Sensitivity
: The icon to the right of the following three settings will enable or disable tablet pressure sensitivity to control how strong the effect is.

Color
: The color of the brush

Radius
: The radius of the brush in pixels

Strength
: How powerful the brush is when applied}}

Blend
: Set the way the paint is applied over the underlying texture

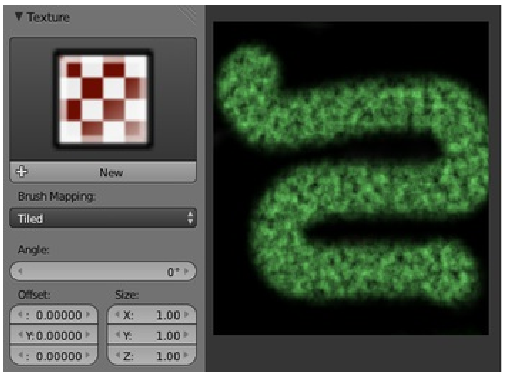
- Mix: the brush color is mixed in with existing colors
- Add: the brush color is added to the existing color; green added to red gives yellow.
- Subtract: the brush color is subtracted; painting blue on purple gives red
- Multiply: the RGB value of the base is multiplied by the brush color
- Lighten: the RGB value of the base color is increased by the brush color
- Darken: tones down the colors
- Erase Alpha: makes the image transparent where painted, allowing background colors and lower-level textures to show through. As you 'paint', the false checkerboard background will be revealed
- Add Alpha: makes the image more opaque where painted

  In order to see the effects of the Erase and Add Alpha mix modes in the UV/Image Editor, you must enable the alpha channel display by clicking the Display Alpha or the Alpha-Only button. Transparent (no alpha) areas will then show a checkered background.

Image
: When using the clone brush, this allows you to select an image as a clone source.

Alpha
: Opacity of the clone image display

Texture



Brush Mapping: Tiled
Angle: 0°
Offset: 0.00000, Y: 0.00000, 0.00000
Size: X: 1.00, Y: 1.00, Z: 1.00

Texture options and example

Use the texture selector at the bottom of the paint panel to select a pre-loaded image or procedural texture to use as your brush pattern. Note that in order to use it, you must have a placeholder material defined, and that particular texture defined using the Material and Texture buttons. It is not necessary to have that material or texture applied to any mesh anywhere; it must only be defined. The example to the right shows the effects of painting with a flat (banded) wood texture. Switching the texture to Rings makes a target/flower type of brush painting pattern.

Note: In Clone paint mode, this field changes to indicate the picture image or texture that you are cloning from.

Brush Mapping
: Sets how the texture is applied to the brush

  View Plane

  : In 2D painting, the texture moves with the brush

  Tiled

  : The texture is offset by the brush location

  3D

  : Same as tiled mode

  Stencil

  : Texture is applied only in borders of the stencil.

  Random

  : Random applying of texture.

Angle
: This is the rotation angle of the texture brush. It can be changed interactively via CtrlF in the 3D view. While in the interactive rotation you can enter a value numerically as well. Can be set to:

  User
  : Directly input the angle value.

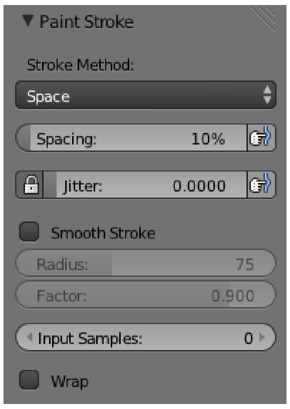
  Rake
  : Angle follows the direction of the brush stroke. Not available with 3D textures.

  Random
  : Angle is randomized.

Offset
: Offset the texture in x, y, and z.

Size
: Set the scale of the texture in each axis.

Stroke



Stroke Method:
Space
Spacing: 10%
Jitter: 0.0000
Smooth Stroke
Radius: 75
Factor: 0.900
Input Samples: 0
Wrap

Stroke panel

Stroke Method
: Allows set the way applying strokes.

  Airbrush
  : Flow of the brush continues as long as the mouse click is held, determined by the Rate setting. If disabled, the brush only modifies the color when the brush changes its location.

    Rate
    : Interval between paints for airbrush

  Space
  : Creates brush stroke as a series of dots, whose spacing is determined by the Spacing setting.

    Spacing
    : Represents the percentage of the brush diameter. Limit brush application to the distance specified by spacing.

  Dots
  : Apply paint on each mouse move step

Jitter
Jitter the position of the brush while painting

Smooth stroke
Brush lags behind mouse and follows a smoother path. When enabled, the following become active:

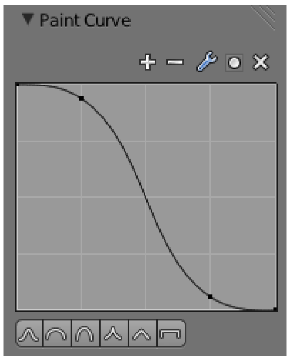
Radius
Sets the minimun distance from the last point before stroke continues.

Factor
Sets the amount of smoothing.

Input Samples
Average multiple input samples together to smooth the brush stroke.

Wrap
wraps your paint to the other side of the image as your brush moves off the OTHER side of the canvas (any side, top/bottom, left/right). Very handy for making seamless textures.

Curve

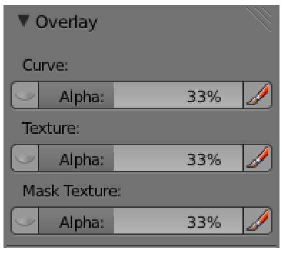


Curve panel

The paint curve allows you to control the falloff of the brush. Changing the shape of the curve will make the brush softer or harder.

Paint options

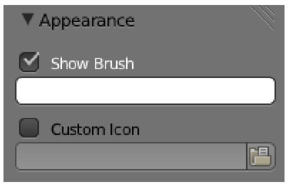
Overlay



Overlay panel

Allows you to customize the display of curve and texture that applied to the brush.

Appearance



Appearance panel

Allows you to customize the color of the brush radius outline, as well as specify a custom icon.

Saving

If the header menu item Image has an asterisk next to it, it means that the image has been changed, but not saved. Use the Image->Save Image option to save your work with a different name or overwrite the original image.

UV Textures
Since images used as UV Textures are functionally different from other images, you should keep them in a directory separate from other images.

The image format for saving is independent of the format for rendering. The format for saving a UV image is selected in the header of the Save Image window, and defaults to PNG (.png).

If Packing is enabled in the window header, or if you manually Image->Pack Image, saving your images to a separate file is not necessary.

Using an External Image Editor

If you use an external program to edit your UV Texture, you must:

1. run that paint program (GIMP, Photoshop, etc.)
2. load the image or create a new one
3. change the image, and
4. re-save it within that program.
5. Back in Blender, you reload the image in the UV/Image Editor window.

You want to use an external program if you have teams of people using different programs that are developing the UV textures, or if you want to apply any special effects that Texture Paint does not feature, or if you are much more familiar with your favorite paint program.

Projection Texture Painting

Projection texture painting allows an artist to paint on texture mapped on a 3D model. Unlike painting in the image editor, projection texture painting is done in the 3D viewport of blender.

Getting Started

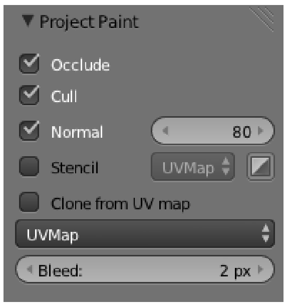
To enter texture paint mode, you need to select a mesh object and select *Texture Paint* from the mode menu (the one which toggles between *Object*, *Edit* etc. modes).

Painting on a 3D model requires some setup before being possible. Blender needs a way to map an image to the 3D model. This is accomplished by using a UV map (see UV Mapping for more details), so if the model hasn't been unwrapped yet, it should be unwrapped prior to entering *Texture Paint* mode. The image assigned to the UV layer is also used for painting. That means that the user should either:

- unwrap the model while the target image is being displayed in the image editor window, or
- unwrap, and while still in edit mode, change the image in the UV editor window to the target image.

If the target image is not square, the first method is preferable, so that unwrapping accounts for the aspect ratio of the image.

Hints

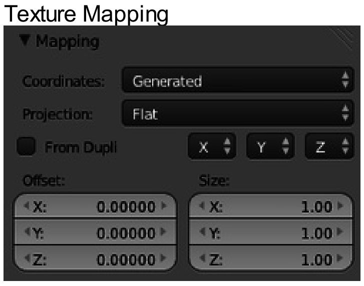


Project Paint panel

There are a known limitations in painting...

- Overlapping UVs are not supported (as with texture baking).
- When painting onto a face which is partially behind the view (in perspective mode), the face can't be painted on. To avoid, this zoom out or use an Ortho mode viewport.
- When painting onto a face in perspective mode onto a low poly object with normals pointing away from the view, painting may fail; to workaround disable the **Normal** option in the paint panel.
  Typically this happens when painting onto the side of a cube T34665

Texture Mapping

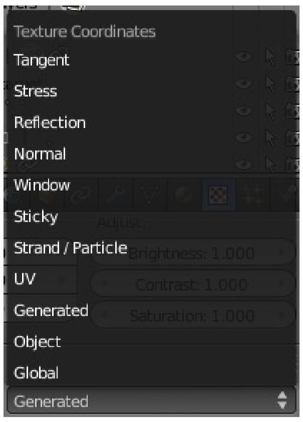


Mapping panel

Textures need mapping coordinates, to determine how they are applied to the object. The mapping specifies how the texture will ultimately wrap itself to the object.

For example, a 2D image texture could be configured to wrap itself around a cylindrical shaped object.

Coordinates



Mapping Coordinate menu

Coordinates Mapping works by using a set of coordinates to guide the mapping process. These coordinates can come from anywhere, usually the object to which the texture is being applied to.

Global
: The scene's global 3D coordinates. This is also useful for animations; if you move the object, the texture moves across it. It can be useful for letting objects appear or disappear at a certain position in space.

Object
: Uses an object as source for coordinates. Often used with an Empty, this is an easy way to place a small image at a given point on the object (see the example below). This object can also be animated, to move a texture around or through a surface.

  Object
  : Select the name of an object.

Generated
: The original undeformed coordinates of the object. This is the default option for mapping textures.

UV
: UV mapping is a very precise way of mapping a 2D texture to a 3D surface. Each vertex of a mesh has its own UV co-ordinates which can be unwrapped and laid flat like a skin. You can almost think of UV coordinates as a mapping that works on a 2D plane with its own local coordinate system to the plane on which it is operating on. This mapping is especially useful when using 2D images as textures, as seen in UV Mapping. You can use multiple textures with one set of UV coordinates.

  Layer
  : Select your UV layer to use it for mapping.

Strand/Particle
: Uses normalized 1D strand texture coordinate or particle age(X) and trail position (Y). Use when texture is applied to hair strands or particles.

Sticky
: Uses a mesh's sticky coordinates, which are a form of per-vertex UV co-ordinates. If you have made sticky coordinates first (in (usually) Camera View → Space → type Sticky → choose Add Sticky/Remove Sticky), the texture can be rendered in camera view (so called "Camera Mapping").

Window
: The rendered image window coordinates. This is well suited to blending two objects.

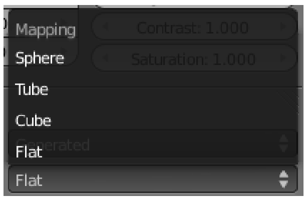
Normal
: Uses the direction of the surface's normal vector as coordinates. This is very useful when creating certain special effects that depend on viewing angle.

Reflection
: Uses the direction of the reflection vector as coordinates. This is useful for adding reflection maps — you will need this input when Environment Mapping.

Stress
: Uses the difference of edge length compared to original coordinates of the mesh. This is useful, for example, when a mesh is deformed by modifiers.

Tangent
: Uses the optional tangent vector as texture coordinates.

Projection



Projection menu

Flat
: Flat mapping gives the best results on single planar faces. It does produce interesting effects on the sphere, but compared to a sphere-mapped sphere the result looks flat. On faces that are not in the mapping plane the last pixel of the texture is extended, which produces stripes on the cube and cylinder.

Cube
: Cube mapping often gives the most useful results when the objects are not too curvy and organic (notice the seams on the sphere).

Tube
: Tube mapping maps the texture around an object like a label on a bottle. The texture is therefore more stretched on the cylinder. This mapping is of course very good for making the label on a bottle or assigning stickers to rounded objects. However, this is not a cylindrical mapping so the ends of the cylinder are undefined.

Sphere
: Sphere mapping is the best type for mapping a sphere, and it is perfect for making planets and similar objects. It is often very useful for creating organic objects. It also produces interesting effects on a cylinder.

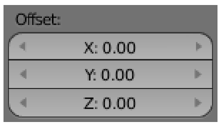
Inheriting coordinates from the parent object

From Dupli

Duplis instanced from vertices, faces, or particles, inherit texture coordinates from their parent.

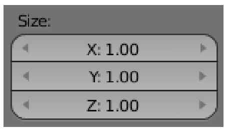
Todo: explaination

Coordinate Offset, Scaling and Transformation



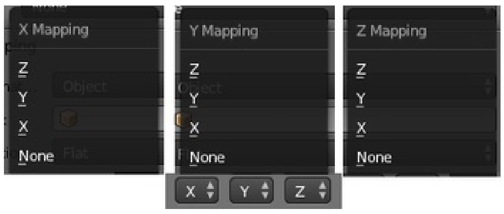
Offset panel

Offset
: The texture co-ordinates can be translated by an offset. Enlarging of the Ofs moves the texture towards the top left.



Size panel

Size
: Allows scaling of the texture coordinates.



Mapping axes

X, Y and Z Mapping
These buttons allow you to change the mapping of axes between the texture's own coordinate system, and the mapping system you choose (Generated, UV, etc.) More precisely, to each axis of the texture corresponds one of four choices, that allow you to select to which axis in the mapping system it maps! This implies several points:

- For 2D textures (such as images), only the first two rows are relevant, as they have no Z data.
- You can rotate a 2D picture a quarter turn by setting the first row (i.e. X texture axis) to Y, and the second row (Y texture axis) to X.
- When you map no texture axis (i.e. the three "void" buttons are set), you'll get a solid uniform texture, as you use zero dimension (i.e. a dot, or pixel) of it (and then Blender extends or repeats this point's color along all axes.)
- When you only map one texture axis (i.e. two "void" buttons are enabled), you'll get a "striped" texture, as you only use one dimension (i.e. a line of pixel) of it, and then Blender stretches this line along the two other axes.
- The same goes, for 3D textures (i.e. procedural ones), when one axis is mapped to nothing, Blender extends the plan ("slice") along the relevant third axis.

So, all this is a bit hard to understand and master. Fortunately, you do not have to change these settings often, except for some special effects… Anyway, the only way to get used to them is to practice!

Environment Maps

Environment maps take a render of the 3D scene and apply it to a texture, to use for faking reflections. If you want to achieve a very realistic result, raytraced reflections are a good solution. Environment Maps are another way to create reflective surfaces, but they are not so simple to set up.

So why should one use Environment Maps?

- The main reason is probably that they can be much faster than raytracing reflections. In certain situations they need to be calculated only once, and may be reused like any ordinary texture. You may even modify the precalculated Environment Map in an image editor.
- Environment maps can also be blurred and render even faster because the resolution can then be lowered. Blurring a reflection with the raytracer always adds to the render time, sometimes quite a lot.
- Halos (a visualization type for particles) are not visible to raytraced reflections, so you need to setup environment maps to reflect them.
- Keypoint strands (another visualization type for particles) are also not visible to raytraced reflections, so you need to setup environment maps to reflect them.

Just as we render the light that reaches the viewing plane using the camera to define a viewpoint, we can render the light that reaches the surface of an object (and hence, the light that might ultimately be reflected to the camera). Blender's environment mapping renders a cubic image map of the scene in the six cardinal directions from any point. When the six tiles of the image are mapped onto an object using the Refl input coordinates, they create the visual complexity that the eye expects to see from shiny reflections.

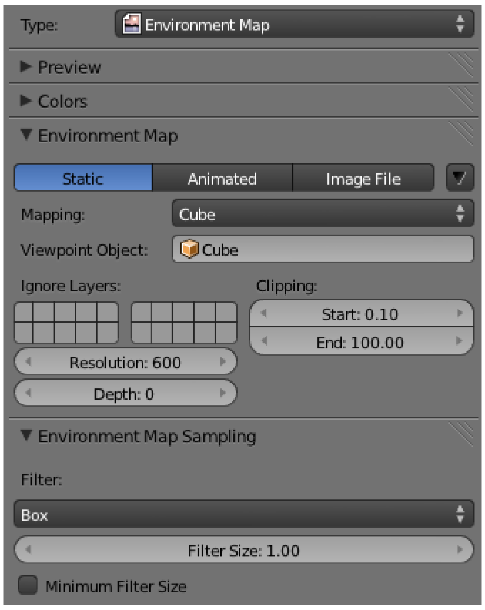
Note

It's useful to remember here that the true goal of this technique is *believability*, not *accuracy*. The eye doesn't need a physically accurate simulation of the light's travel; it just needs to be lulled into believing that the scene is real by seeing the complexity it expects. The most unbelievable thing about most rendered images is the sterility, not the inaccuracy.

Options

Important

For correct results, the mapping of an environment map texture must be set to 'Refl' (reflection co-ordinates) in the Map Input panel of the Material context.


Type: Environment Map
Environment Map: Static
Mapping: Cube
Viewpoint Object: Cube
Clipping: Start: 0.10, End: 100.00
Resolution: 600
Depth: 0
Filter: Box
Filter Size: 1.00
Minimum Filter Size

Reflecting plane EnvMap settings.

Blender allows three types of environment maps, as you can see in *Reflecting plane EnvMap settings.*:

Static
: The map is only calculated once during an animation or after loading a file.

Animated
: The map is calculated each time a rendering takes place. This means moving Objects are displayed correctly in mirroring surfaces.

Image File
: When saved as an image file, environment maps can be loaded from disk. This option allows the fastest rendering with environment maps, and also gives the ability to modify or use the environment map in an external application.

  When using planar reflections, if the camera is the only moving object and you have a reflecting plane, the Empty must move too and you must use Anim environment map. If the reflecting object is small and the Empty is in its center, the environment map can be Static, even if the object itself rotates since the Empty does not move. If, on the other hand, the Object translates the Empty should follow it and the environment map be of Anim type.

Options in dropdown menu:

Clear Environment Map
: Clears the currently rendered environment map from memory. This is useful to refresh a Static environment maps and you have changed things in your scene since the last time the environment map was rendered. Anim environment maps do this automatically on every render.

Save Environment Map
: Saves the currently stored static environment map to disk as an image file. This can be loaded again with Load.

Clear All Environment Maps
: Does the same as Free Data, but with all environment maps in the scene. This is a useful shortcut when using recursive environment maps (when the Depth is greater than 0).

Note

EnvMap calculation can be disabled at a global level by the EnvMap Tog Button in the Render Panel of the Rendering Buttons.

Viewpoint Object
: Environment maps are created from the perspective of a specified object. The location of this object will determine how 'correct' the reflection looks, though different locations are needed for different reflecting surfaces. Usually, an Empty is used as this object.

- For planar reflections, the object should be in a location mirrored from the camera, on the other side of the plane of reflection (see Examples). This is the most accurate usage of Environment maps.
- For spherical reflections, the object should be in the center of the sphere. Generally, if the reflecting sphere's object center point is in the center of its vertices, you can just use the name of the actual sphere object as the Ob:
- For irregular reflections, there's no hard and fast rule, you will probably need to experiment and hope that the inaccuracy doesn't matter.

Ignore Layers
: The layers to exclude from the environment map creation. Since environment maps work by rendering the scene from the location of the Ob: object, you will need to exclude the actual reflecting surface from the environment map, otherwise it will occlude other objects that should be reflected on the surface itself.

Eg. If you are rendering an environment map from the center of a sphere, all the environment map will show by default is the inside of the sphere. You will need to move the sphere to a separate layer, then exclude that layer from the environment map render, so that the environment map will show (and hence reflect) all the objects outside the sphere.

Resolution
: The resolution of the cubic environment map render. Higher resolutions will give a sharper texture (reflection), but will be slower to render.

Depth
: The number of recursive environment map renders. If there are multiple reflecting objects using environment maps in the scene, some may appear solid, as they won't render each other's reflections. In order to show reflections within reflections, the environment maps need to be made multiple times, recursively, so that the effects of one environment map can be seen in another environment map. See Examples.

Clipping Start/End
: The clipping boundaries of the virtual camera when rendering the environment map. Sets the minimum and maximum distance from the camera that will be visible in the map.

Environment Map Sampling

Filter

Box

Box Filter

EWA

Elliptical Weighted Average — one of the most efficient direct convolution algorithms developed by Paul Heckbert and Ned Greene in the 1980s. For each texel, EWA samples, weights, and accumulates texels within an elliptical footprint and then divides the result by the sum of the weights.

Eccentricity

Maximum eccentricity (higher gives less blur at distant/oblique angles, but is also slower)

FELINE

FELINE (Fast Elliptical Lines), uses several isotropic probes at several points along a line in texture space to produce an anisotropic filter to reduce aliasing artifacts without considerably increasing rendering time.

Probes

Maximum number of samples (higher gives less blur at distant/oblique angles, but is also slower)

Area

Area filter to use for image sampling.

Eccentricity

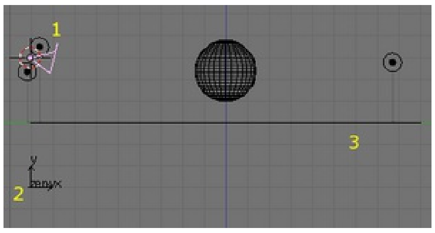
Maximum eccentricity (higher gives less blur at distant/oblique angles, but is also slower)

Filter Size
: The amount of blurring applied to the texture. Higher values will blur the environment map to fake blurry reflections.

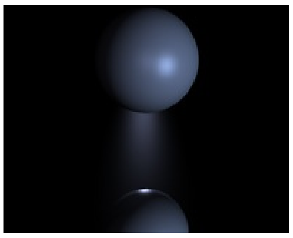
Minimum Filter Size
: Use Filter Size as a minimal filter value in pixels

Examples

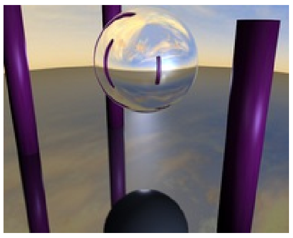
In this example, an empty is used as the Ob: of the reflecting plane's environment map. It is located in the specular position of the camera with respect to the reflecting surface. (This is possible, strictly speaking, only for planar reflecting surfaces.) Ideally, the location of the empty would mirror the location of the camera across the plane of the polygon onto which it is being mapped.

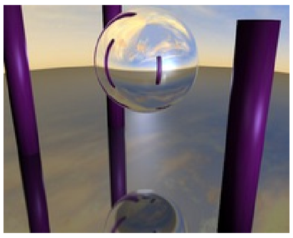


Planar reflection example. 1: Camera, 2: Empty, 3: Reflecting Plane.


Sphere on a reflecting surface.

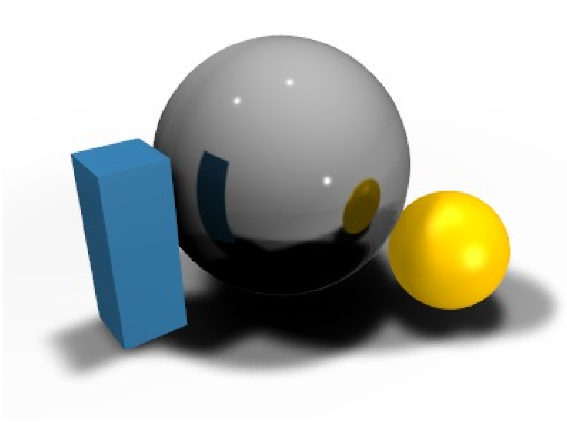
The following images show the effect of the Depth. The first render has depth set to 0. This means the environment map on the plane has rendered before the environment map of the sphere, so the sphere's reflection isn't shown. By raising the Depth, the environment map is rendered recursively, in order to get reflections of reflections.


Reflecting sphere on a reflecting surface.


Reflecting sphere on a reflecting surface with multiple reflections.

Limitations

Because environment maps are calculated from the exact location of the Viewpoint Object's object center, and not from actual reflecting surface, they can often be inaccurate, especially with spheres. In the following image, the rectangular prism and the smaller spheres are touching the sides of the large reflecting sphere, but because the environment map is calculated from the center of the sphere, the surrounding objects look artificially far away.


Inaccurate spherical reflection, the coloured objects are artificially offset

UV Mapping

The most flexible way of mapping a 2D texture over a 3D object is a process called "UV mapping". In this process, you take your three-dimensional (X,Y & Z) mesh and unwrap it to a flat two-dimensional (X & Y ... or rather, as we shall soon see, "U & V") image. Colors in the image are thus mapped to your mesh, and show up as the color of the faces of the mesh. Use UV texturing to provide realism to your objects that procedural materials and textures cannot do, and better details than Vertex Painting can provide.

UVs Explained

Box being inspected

Box mapped flat

The best analogy to understanding UV mapping is cutting up a cardboard box. The box is a three-dimensional (3D) object, just like the mesh cube you add to your scene.

If you were to take a pair of scissors and cut a seam or fold of the box, you would be able to lay it flat on a tabletop. As you are looking down at the box on the table, we could say that U is the left-right direction, is V is the up-down direction. This image is thus in two dimensions (2D). We use **U** and **V** to refer to these "texture-space coordinates" instead of the normal **X** and **Y**, which are always used (along with **Z**) to refer to "3D space."

When the box is reassembled, a certain UV location on the paper is transferred to an (X,Y,Z) location on the box. This is what the computer does with a 2D image in wrapping it around a 3D object.

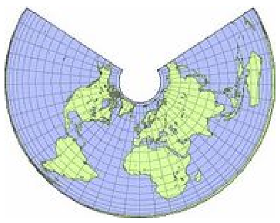
During the UV unwrapping process, you tell Blender exactly how to map the faces of your object (in this case, a box) to a flat image in the UV/Image Editor window. You have complete freedom in how to do this. (Continuing our previous example, imagine that, having initially laid the box flat on the tabletop, you now cut it into smaller pieces, somehow stretch and/or shrink those pieces, and then arrange them in some way upon a photograph that's also lying on that tabletop ...)

Cartography Example

Cartographers (map makers) have been dealing with this problem for millennia. A cartography (map-making) example is creating a projection map of the whole world. In cartography, we take the surface of the earth (a sphere) and make a flat map that can be folded up into the glove compartment aboard the space shuttle. We 'fill in' spaces toward the poles, or change the outline of the map in any of several ways:

Mercator Projection

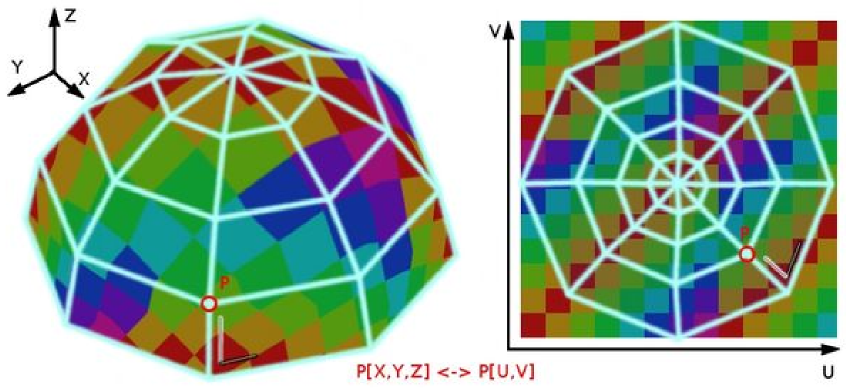
Mollweide Projection


Albers-equal Projection

Each of these is an example of a way to UV map a sphere. Each of the hundred or so commonly accepted projections has its advantages and disadvantages. Blender allows us to do the same thing any way we want to, on the computer.

On more complex models (like seen in the earth map above) there pops up an issue where the faces can't be 'cut', but instead they are stretched in order to make them flat. This helps making easier UV maps, but sometimes adds distortion to the final mapped texture. (Countries and states that are closer to the North or the South Pole look smaller on a flat map than do ones which are close to the Equator.)

Half-Sphere Example



3D Space (XYZ) versus UV Space (click to enlarge)

In this image you can easily see that the shape and size of the marked face in 3D space is different in UV space.

This difference is caused by the 'stretching' (technically called mapping) of the 3D part (XYZ) onto a 2D plane (i.e the UV map).

If a 3D object has a UV map, then, in addition to the 3D-coordinates X, Y, and Z, each point on the object will have corresponding U and V coordinates. (P in the image above is an example of how a point on a 3D object might be mapped onto a 2D image.)

The UV Editor

About fuctionalities for mapping UV see UV/Image Editor section for details.

Advantages of UVs

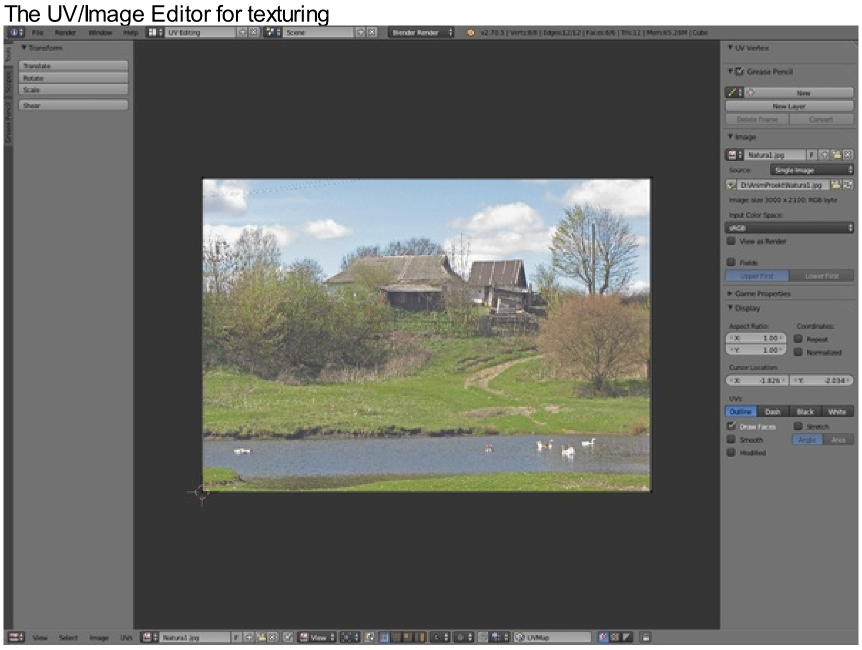
While procedural textures (described in the previous chapters) are useful-they never repeat themselves and always "fit" 3D objects-they are not sufficient for more complex or natural objects. For instance, the skin on a human head will never look quite right when procedurally generated. Wrinkles on a human head, or scratches on a car do not occur in random places, but depend on the shape of the model and its usage. Manually-painted images, or images captured from the real world gives more control and realism. For details such as book covers, tapestry, rugs, stains, and detailed props, artists are able to control every pixel on the surface using a UV Texture.

A UV map describes what part of the texture should be attached to each polygon in the model. Each polygon's vertex gets assigned to 2D coordinates that define which part of the image gets mapped. These 2D coordinates are called UVs (compare this to the XYZ coordinates in 3D). The operation of generating these UV maps is also called "unwrap", since it is as if the mesh were unfolded onto a 2D plane.

For most simple 3D models, Blender has an automatic set of unwrapping algorithms that you can easily apply. For more complex 3D models, regular Cubic, Cylindrical or Spherical mapping, is usually not sufficient. For even and accurate projection, use seams to guide the UV mapping. This can be used to apply textures to arbitrary and complex shapes, like human heads or animals. Often these textures are painted images, created in image editing and manipulation software like Gimp, Photoshop, etc.

Games
UV mapping is also essential in the Blender game engine, or any other game. It is the de facto standard for applying textures to models; almost any model you find in a game is UV mapped.

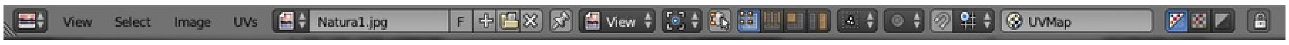
The UV/Image Editor for texturing



UV/Image Editor window for texturing

The UV/Image Editor is where you will be editing the UVs. This is an overview of the tools found there. Using the UV editor is explained more in depth in the next sections.

Header Bar



UV/Image Editor Header

The header bar contains several menus and options for working with UVs

View menu
: Tools for Navigating, working with the editor and controlling how things are displayed. The properties panel has display options and manipulation tools. When an image is being used, image properties are displayed. The Scopes panel is used when working with Images. It contains different image visualizers

Select menu
: Tools for Selecting UVs.

Image menu
: This contains options for when Working with Images and Painting Textures.

UVs menu
: Contains tools for Unwrapping Meshes and Editing Uvs.

Image Selector Menu
: Select the image to apply when Working with Images.

Pin Image
: Displays current image regardless of selected object.

Pivot Point Selector
: Similar to working with Pivot Points in the 3D view.

Sync Selection
: Keeps UV and Mesh component selections in sync.

Selection Modes

- Vertex
- Edge
- Face
- Island

Sticky Selection Mode
: When Sync Selection is disabled, these options control how UVs are selected.

Proportional Editing
: Works like Proportional Editing in the 3d view

UV Snapping
: Similar to Snapping in the 3D View

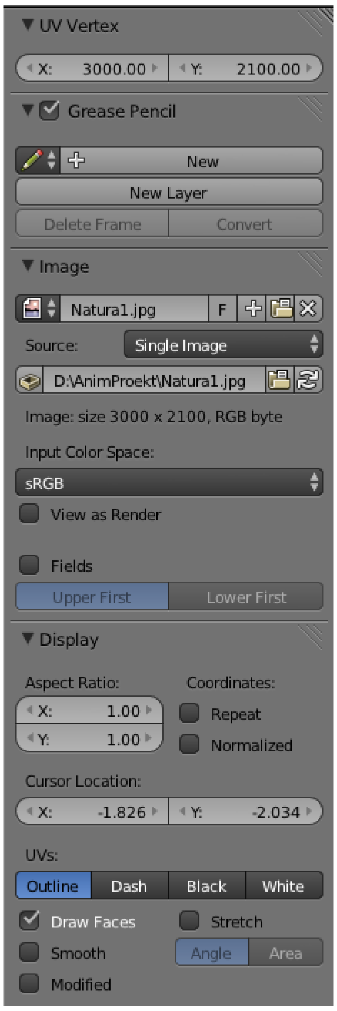
Active UV Texture Map Selector
: Select which UV texture to use

Image Channels to Draw

Set the image to be displayed with Color, Color and Alpha, or just Alpha.

Auto Update Other Affected Windows
: Update other affected windows space automatically to reflect changes during interactive operations such as transfom.

Properties Panel



X: 3000.00 Y: 2100.00

Grease Pencil ✓

Natural1.jpg

Source: Single Image

D:\AnimProekt\Natural1.jpg

Image: size 3000 x 2100, RGB byte

Input Color Space: sRGB

View as Render

Fields

Upper First Lower First

Aspect Ratio: X: 1.00 Y: 1.00

Repeat

Normalized

Cursor Location: X: -1.826 Y: -2.034

✓ Draw Faces

Stretch

Smooth

Modified

UV/Image Editor Properties panel

UV Vertex
: Transform Properties for select UVs

Grease Pencil
: Similar to Grease Pencil in the 3d view.

Image
: Contains the properties of the current Image

Display
: Controls display options for UVs and additional settings for when Working with Images.

Display Options

You can set how UVs are displayed in the Display Panel:

Aspect Ratio
: Display Aspect for this image. Does not affect rendering.

Coordinates
: Display UV coordinates

  Repeat
  : Draw the image repeated outside of the main view.

  Normalized
  : Display UV coordinates from 0.0 to 1.0 rather then in pixels

Cursor Location
: 2D cursor location for this view

Outline/Dash/Black/White
: Sets how UV edges are displayed

Draw Faces
: Draw faces over the image

Smooth
: Makes edges appeared Antialiased

Modified
: Show results of modifiers in the UV display

Stretch
: Shows how much of a difference there is between UV coordinates and 3D coordinates. Blue means low distortion, while Red means high distortion. Choose to display the distortion of Angles or the Area.

Navigating in UV Space

Panning can be done by clicking the MMB and dragging.

Zooming can be done by scrolling MMB up or down. Also, as in the 3D view, you can use + NumPad or - NumPad to zoom.

The following shortcuts are available, and through the View Menu:

- Zoom 1:8 8 NumPad
- Zoom 1:4 4 NumPad
- Zoom 1:2 2 NumPad
- Zoom 1:1 1 NumPad
- Zoom 2:1 ⇧ Shift2 NumPad
- Zoom 4:1 ⇧ Shift4 NumPad
- Zoom 8:1 ⇧ Shift8 NumPad

- View All ↖ Home
- View Center . NumPad

UV Mapping a Mesh

The first step is to unwrap your mesh. You want to unwrap when you feel your mesh is complete with respect to the number of faces it needs to have. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you, but you may need to do additional mapping or editing. In this fashion, you can use the UV Texture image to guide additional geometry changes.

This section covers techniques for Mapping Uvs. The next sections cover Editing UVs, followed by methods of Managing UV Layouts, and Applying Images to UVs.

About UVs

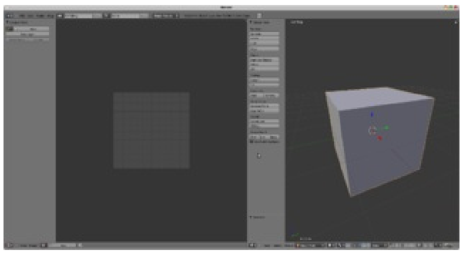
Every point in the UV map corresponds to a vertex in the mesh. The lines joining the UVs correspond to edges in the mesh. Each face in the UV map corresponds to a mesh face.

Each face of a mesh can have many UV Textures. Each UV Texture can have an individual image assigned to it. When you unwrap a face to a UV Texture in the UV/Image Editor, each face of the mesh is automatically assigned *four UV coordinates:* These coordinates define the way an image or a texture is mapped onto the face. These are 2D coordinates, which is why they're called UV, to distinguish them from XYZ coordinates. These coordinates can be used for rendering or for real-time OpenGL display as well.

Every face in Blender can have a link to a different image. The UV coordinates define how this image is mapped onto the face. This image then can be rendered or displayed in real time. A 3D window has to be in "Face Select" mode to be able to assign Images or change UV coordinates of the active Mesh Object. This allows a face to participate in many UV Textures. A face at the hairline of a character might participate in the facial UV Texture, *and* in the scalp/hair UV Texture.

These are described more fully in the next sections.

Getting Started

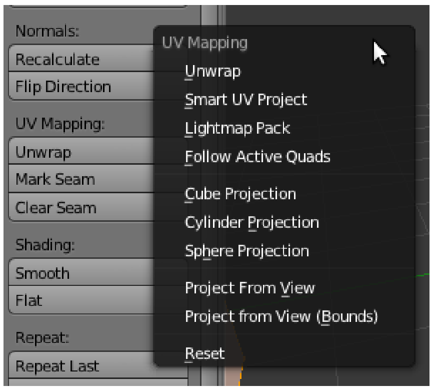

UV Editing screen layout

By default, meshes are not created with UVs. First you must map the faces, then you can edit them. The process of unwrapping your model is done within Edit Mode in the 3D View window. This process creates one or more UV Islands in the UV/Image Editor window.

To begin, choose the UV Editing screen layout from the selection list at the top of your screen in the User Preferences window header. This sets one of the panes to show you the UV/Image Editor window (⇧ ShiftF10), and the other pane to the 3D window (⇧ ShiftF5).

Enter edit mode, as all unwrapping is done in Edit mode. You can be in vertex, face, or edge selection mode.

Workflow



Choosing the unwrapping method

The process for unwrapping is straightforward, but there are tons of options available, each of which dramatically affect the outcome of the unwrap. By understanding the meaning behind the options, you will become more efficient at unwrapping. The process is:

1. Mark Seams if necessary
2. Select all of the mesh components
3. Select a UV mapping method from the UV Unwrap menu
4. Adjust the unwrap settings
5. Add a test image to see if there will be any distortion. See Applying Images to UVs
6. Adjust UVs in the UV editor. See Editing UVs

Mapping Types

Blender offers several ways of mapping UVs. The simpler projection methods use formulas that map 3d space onto 2d space, by interpolating the position of points toward a point/axis/plane through a surface. The more advanced methods can be used with more complex models, and have more specific uses.

Basic:

Cube
: Maps the mesh onto the faces of a cube, which is then unfolded.

Sphere
: Projects the UVs onto a spherical shape. Useful only for spheres or spherical shapes, like eyes, planets, etc.

Cylinder
: Projects UVs onto a cylindrical surface.

Project from View
: Takes the current view in the 3D viewport and flattens it as it appears.

Advanced:

Unwrap
: Useful for organic shapes. Smooths the mesh into a flat surface by cutting along seams.

Smart UV Project
: Breaks the mesh into islands based on an angle threshold.

Lightmap Pack
: Separates each face and packs them onto the UV grid.

Follow Active Quads
: Follow UV from active quads along continuous face loops.

You can also reset UVs, which maps each face to fill the UV grid, giving each face the same mapping.

If we were to use an image that was tileable, the surface would be covered in a smooth repetition of that image, with the image skewed to fit the shape of each individual face. Use this unwrapping option to reset the map and undo any unwrapping (go back to the start).

Basic Mapping

Based on the fundamental geometry of the object, and how it is being viewed, the Mesh->UV Unwrap->Cube, Cylinder, and Sphere UV Calculations attempt to unfold the faces for you as an initial best fit. Here, the view from the 3D window is especially important. Also, the settings for cube size or cylinder radius (Editing buttons, UV Calculation panel) should be set (in Blender units) to encompass the object.

The following settings are common for the Cube, Cylinder, and Sphere mappings:

Correct Aspect
: Map UVs taking image aspect ratios into consideration. If an image has already been mapped to the texture space that is non-square, the projection will take this into account and distort the mapping to appear correct.

Clip to Bounds
: Any UVs that lie outside the 0 to 1 range will be clipped to that range by being moved to the UV space border it is closest to.

Scale to Bounds
: If the UV map is larger than the 0 to 1 range, the entire map will be scaled to fit inside.

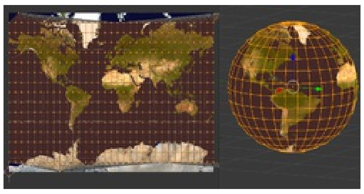
Cube

Cube mapping projects s mesh onto six separate planes, creating 6 UV islands. In the UV editor, these will appear overlapped, but can be moved. See Editing UVs.

Cube Size

: Set the size of the cube to be projected onto.

Cylinder and Sphere


Using a Mercator image with a Sphere Projection

Cylindrical and Spherical mappings have the same settings. The difference is that a cylindrical mapping projects the UVs on a plan toward the cylinder shape, while a spherical map takes into account the sphere’s curvature, and each latitude line becomes evenly spaced.

Normally, to unwrap a cylinder (tube) as if you slit it lengthwise and folded it flat, Blender wants the view to be vertical, with the tube standing 'up'. Different views will project the tube onto the UV map differently, skewing the image if used. However you can set the axis on which the calculation is done manually. This same idea works for the sphere mapping:

Recall the opening cartographer's approaching to mapping the world? Well, you can achieve the same here when unwrapping a sphere from different perspectives. Normally, to unwrap a sphere, view the sphere with the poles at the top and bottom. After unwrapping, Blender will give you a Mercator projection; the point at the equator facing you will be in the middle of the image. A polar view will give a very different but common projection map. Using a Mercator projection map of the earth as the UV image will give a very nice planet mapping onto the sphere.

Direction
- View on Poles
  - Use when viewing from the top (at a pole) by using an axis that is straight down from the view
- View on Equator
  - Use if view is looking at the equator, by using a vertical axis
- Align to Object
  - Uses the object's transform to calculate the axis

Align
- Select which axis is up
- Polar ZX
  - Polar 0 is on the x axis
- Polar ZY
  - Polar 0 is on the y axis

Radius
- The radius of the cylinder to use

Project From View

In the 3D window, the Face->Unwrap UVs->Project from View option maps the face as seen through the view of the 3D window it was selected from. It is almost like you had x-ray vision or squashed the mesh flat as a pancake onto the UV map. Use this option if you are using a picture of a real object as a UV Texture for an object that you have modeled. You will get some stretching in areas where the model recedes away from you.

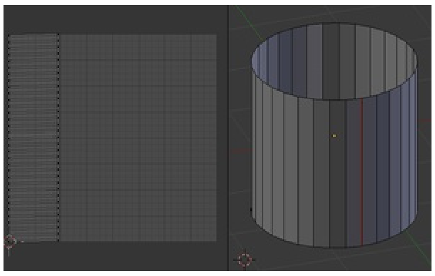
Using Project from View (Bounds) will do the same as above, but scales the UVs to the bounds of the UV space.

Resetting UVs

In the 3D window, Face->Unwrap->Reset maps each selected face to the same area of the image, as previously discussed. To map all the faces of an object (a cube, for example) to the same image, select all the faces of the cube, and unwrap them using the Reset menu option.

Advanced Mapping

Unwrapping Using Seams


Simple Seam on a Cylinder

For many cases, using the Unwrap calculations of Cube, Cylinder, Sphere, or best fit will produce a good UV layout. However, for more complex meshes, especially those with lots of indentations, you may want to define a **seam** to limit and guide any of the unwrapping processes discussed above.

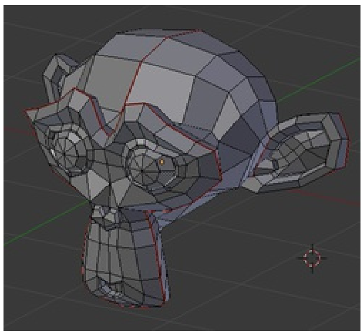
Just like in sewing, a seam is where the ends of the image/cloth are sewn together. In unwrapping, the mesh is unwrapped at the seams. Think of this method as peeling an orange or skinning an animal. You make a series of cuts in the skin, then peel it off. You could then flatten it out, applying some amount of stretching. These cuts are the same as seams.

When using this method, you need to be aware of how much stretching there is. The more seams there are, the less stretching there is, but this is often an issue for the texturing process. It’s a good idea to have as few seams as possible while having the least amount of stretching. Try to hide seams where they will not be seen. In productions where 3d Paint is used, this becomes less of an issue, as projection painting can easily deal with seams, as opposed to 2d texturing, where it is difficult to match the edges of different UV islands.

The workflow is the following:

1. Create seams. A seam is marked in Edit mode by selecting edges to make the seam and then issuing the command to Mark Seam.
2. Unwrap
3. Adjust seams and repeat
4. Manually adjust UVs. See the next section on Editing UVs.

Marking Seams


Seamed Suzanne

To add an edge to a seam, simply select the edge and CtrlE Mark Seam. To take an edge out of a seam, select it, CtrlE and Clear Seam.

In the example to the right, the back-most edge of the cylinder was selected as the seam (to hide the seam), and the default unwrap calculation was used. In the UV/Image Editor window, you can see that all the faces are nicely unwrapped, just as if you cut the seam with a scissors and spread out the fabric.

When marking seams, you can use the Select->Linked Faces or CtrlL in Face Select Mode to check your work. This menu option selects all faces connected to the selected one, up to a seam. If faces outside your intended seam are selected, you know that your seam is not continuous. You do not need continuous seams, however, as long as they resolve regions that may stretch.

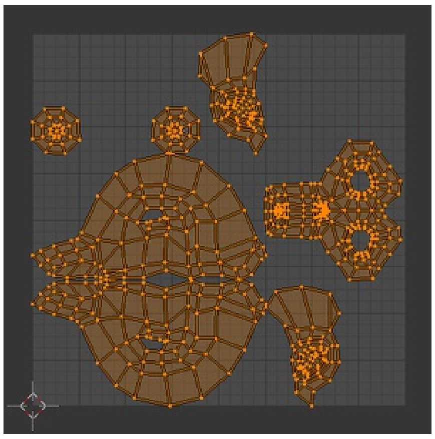
Just as there are many ways to skin a cat, there are many ways to go about deciding where seams should go. In general though, you should think as if you were holding the object in one hand, and a pair of sharp scissors in the other, and you want to cut it apart and spread it on the table with as little tearing as possible. Note that we seamed the outside edges of her ears, to separate the front from the back. Her eyes are disconnected sub-meshes, so they are automatically unwrapped by themselves. A seam runs along the back of her head vertically, so that each side of her head is flattened out.

Another use for seams is to limit the faces unwrapped. For example, when texturing a head, you don't really need to texture the scalp on the top and back of the head since it will be covered in hair. So define a seam at the hairline. Then, when you select a frontal face, and then select linked faces before unwrapping, the select will only go up to the hairline seam, and the scalp will not be unwrapped.

When unwrapping anything that is bilateral, like a head or a body, seam it along the mirror axis. For example, cleave a head or a whole body right down the middle in front view. When you unwrap, you will be able to overlay both halves onto the same texture space, so that the image pixels for the right hand will be shared with the left; the right side of the face will match the left, etc.

Finally, remember that you *don't* have to come up with "one unwrapping that works perfectly for everything everywhere." As we'll discuss later, you can easily have multiple UV unwrappings, using different approaches in different areas of your mesh.

Unwrap


Result of unwrapping Suzanne

Begin by selecting all faces to be unwrapped in the 3D View. With our faces selected, it is now time to unwrap them. In the 3D View, select Mesh->UV Unwrap or U and select Unwrap.

You can also do this from the UV/Image Editor window with command UVs->Unwrap or command E. This method will unwrap all of the faces and reset previous work. The UVs menu will appear in the UV/Image Editor window after unwrapping has been performed once.

The Face->Unwrap->Unwrap option unwraps the faces of the object to provide the 'best fit' scenario based on how the faces are connected and will fit within the image, and takes into account any seams within the selected faces. If possible, each selected face gets its own different area of the image and is not *tucked under* any other faces. If all faces of an object are selected, then each face is mapped to some portion of the image.

Blender has two ways of calculating the unwrapping. They can be selected in the tool setting in the tool panel in the 3D View.

Angle Based
    This method gives a good 2d representation of a mesh.
Conformal
    Uses LSCM (Least Squared Conformal Mapping). This usually gives a less accurate UV mapping than Angle Based, but works better for simpler objects.

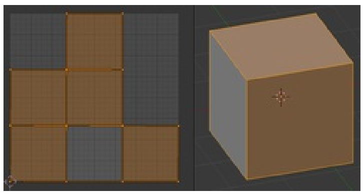
Fill Holes
    Activating Fill Holes will prevent overlapping from occurring and better represent any holes in the UV regions.
Correct Aspect
    Map UVs taking image aspect into account

Use Subsurf Modifier
    Map UVs taking vertex position after subsurf modifier into account

Margin
    Space between UV islands

This point is crucial to understanding mapping later on: a face's UV image texture only has to use *part* of the image, not the *whole* image. Also, portions of the same image can be shared by multiple faces. A face can be mapped to less and less of the total image.

Smart UV Project


Smart UV project on a cube

Smart UV Project, (previously called the Archimapper) gives you fine control over how automatic seams should be created, based on angular changes in your mesh. This method is good for simple and complex geometric forms, such as mechanical objects or architecture.

This function examines the shape of your object, the faces selected and their relation to one another, and creates a UV map based on this information and settings that you supply.

In the example to the right, the Smart Mapper mapped all of the faces of a cube to a neat arrangement of 3 sides on top, 3 sides on the bottom, for all six sides of the cube to fit squarely, just like the faces of the cube.

For more complex mechanical objects, this tool can very quickly and easily create a very logical and straightforward UV layout for you.

The Tool Settings panel in the Tool Shelf allows the fine control over how the mesh is unwrapped:

Angle Limit
    This controls how faces are grouped: a higher limit will lead to many small groups but less distortion, while a lower limit will create fewer groups at the expense of more distortion.

Island Margin
    This controls how closely the UV islands are packed together. A higher number will add more space in between islands.

Area Weight
    Weight projection's vector by faces with larger areas

Lightmap

Lightmap Pack takes each of a mesh's faces, or selected faces, and packs them into the UV bounds. Lightmaps are used primarily in gaming contexts, where lighting information is baked onto texture maps, when it is essential to utilize as much UV space as possible. It can also work on several meshes at once. It has several options that appear in the Tool Shelf:

You can set the tool to map just Selected Faces or All Faces if working with a single mesh.

The Selected Mesh Object option works on multiple meshes. To use this, in Object Mode select several mesh objects, then go into Edit Mode and activate the tool.

Share Tex Space
    This is useful if mapping more than one mesh. It attempts to fit all of the objects' faces in the UV bounds without overlapping.
New UV Layer
    If mapping multiple meshes, this option creates a new UV layer for each mesh. See Managing the Layout.
New Image

Assigns new images for every mesh, but only one if Shared Tex Space is enabled.

Image Size

Set the size of the new image.

Pack Quality
: Pre-packing before the more complex Box packing.

Margin
: This controls how closely the UV islands are packed together. A higher number will add more space in between islands.

Follow Active Quads

The Face->Unwrap->Follow Active Quads takes the selected faces and lays them out by following continuous face loops, even if the mesh face is irregularly shaped. Note that it does not respect the image size, so you may have to scale them all down a bit to fit the image area.

Edge Length Mode:

Even
: Space all UVs evenly.

Length
: Average space UV's edge length of each loop.

Please note that it is the shape of the active quad in UV space that is being followed, not its shape in 3d space. To get a clean 90-degree unwrap make sure the active quad is a rectangle in UV space before using "Follow active quad".

Managing UV Maps

After you finish editing a UV map, you may need to create additional maps on the same object, or transfer a UV map to another mesh.

Transferring UV Maps

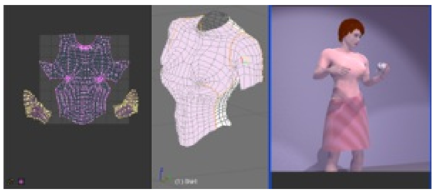
You can copy a UV Map from one mesh to another Mesh provided both meshes have the same geometry/vertex order. This is useful for example when you want to recreate a UV map from an earlier version of your model with intact UVs.

Workflow

- RMB Select the target mesh (to which you want to copy the UV Map)
- ⇧ Shift select the source mesh (that contains the intact UV map)
- Object menu » Make Links... » Transfer UV Layouts (Shortcut: CtrlL ...)

The target Mesh will now have a UV map that matches the original mesh.

Multiple UV Maps

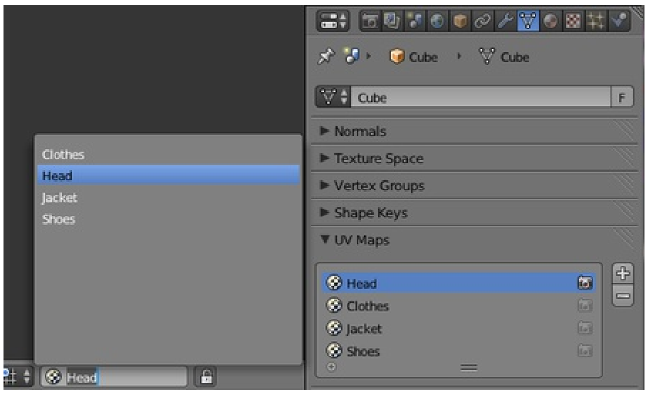

Mesh with Multiple UV Textures

You are not limited to one UV Map per mesh. You can have multiple UV maps for parts of the mesh by creating new UV Textures. The first UV Texture is created for you when you select a face in UV Face Select mode. You can manually create more UV Textures by clicking the New (+) button in the UV maps panel located in the object data button of the properties editor, and unwrapping a different part of the mesh. Those faces will then go with that UV Texture, while the previously unwrapped faces will still go with the previous UV Texture. Note that if you unwrap the same face twice or more times (each time to a different UV Texture), the coloring for that face will be the alpha combination of the layers of those UV Textures.

In the example to the right, we have a mesh for a blouse. The mesh has been seamed as a normal blouse would, shown in the middle in UV Face Select mode. Wishing to make a cut pattern, the front of the blouse was unwrapped and basic rotation and scaling was done to center it in the UV/Image Editor window. It was then moved off to the side, while the left and right sleeves were unwrapped, each rotated and scaled. Then, select a sample face from each cloth piece, in the 3D View Select->Linked Faces, and the UV/Image Editor will show all those pieces (as shown to the right). You can then work with all pieces for that UV Map. The example shows all three pieces moved onto the image area for painting. As you can see, the pattern nicely fits a square yard of cloth.

Another UV Map was created by clicking the New button in the UV Maps panel, and the process was repeated for the backs of the sleeves and the back of the blouse. Two images, one for the front and one for the back, are used to color the fabric. In this case, some faces map to the first texture, while other faces map to the second texture.

UV Textures List



The UV Maps panel (shown to the right) lists the UV Texture maps created for this mesh, and allows you to create New ones as placeholders for future unwrapping operations.

Click the + button to add a new UV texture, and the - to delete an existing one}}. Deleting a UV Map for the mesh destroys all work done in all unwrapping associated the mesh. Click with care. You've been warned.

Each map has a selector button. Click the camera icon to enable that UV texture for rendering. You can change the name by selecting one and changing the text in the Name box. The selected map is displayed in the UV/Image Editor window. The example shows a few UV maps created for a character, and the map for Head is selected.

Note that each texture can be mapped to a specific UV texture. See the Mapping section of the texture panel.

Editing UVs

After unwrap, you will likely need to arrange the UV maps into something that can be logically textured or painted. Your goals for editing are:

- Stitch some pieces (UV maps) back together
- Minimize wasted space in the image
- Enlarge the 'faces' where you want more detail
- Re-size/enlarge the 'faces' that are stretched
- Shrink the 'faces' that are too grainy and have too much detail

With a minimum of dead space, the most pixels can be dedicated to giving the maximum detail and fineness to the UV Texture. A UV face can be as small as a pixel (the little dots that make up an image) or as large as an entire image. You probably want to make some major adjustments first, and then tweak the layout.

Selecting UVs

Selection tools are available in the Select Menu and Header bar, and the shortcuts listed below:

Border Select ; B
: Use the box lasso to select UV coordinates.

Select/Deselect All ; A
: Selects or de-selects all UV coordinates. When initially unwrapping, you will want to select All UVs to rotate, scale, and move them around.

Linked UVsCtrlL
: This menu item selects all UVs that are part of the same UV map. Recall that a map is made for every submesh and seamed part of the mesh, and is analogous to a piece of cloth. Selecting Linked UVs works similarly to the command in 3D View. It will select all UVs that are 'connected' to currently selected UVs.

Pinned UVs ; ⇧ ShiftP
: You can pin UVs so they don't move between multiple unwrap operations. This menu item selects them all. See Pinning

Border Select Pinned ; ⇧ ShiftB
: Use the box lasso to select only pinned UV coordinates.

Unlink Selection ; AltL
: Cuts apart the selected UVs from the map. Only those UVs which belong to fully selected faces remain selected following this command. As the name implies, this is particularly useful to unlink faces and move them elsewhere. The hotkey is analogous to the mesh Separate command.

Selection Modes

Turning on the Sync Selection button in the header causes selection of components in the 3D view to sync with their corresponding elements in the UV editor. This is off by default. These two modes have very different results when transforming components in the UV editor.

When SyncSelection is **Off**: Only selected faces are displayed in the UV editor, and the following selection modes are available:

- Vertex

  Select individual vertices

- Edge

  Select edges

- Face

  Select faces

- Island

  Select contiguous groups of Faces

  The Sticky Selection Mode menu is available in this mode. This controls how UVs are selected:

  Shared Vertex
  : Selects UVs that share a mesh vertex, even if they are in different UV locations.

  Shared Location
  : Selects UVs that are in the same UV location and share a mesh vertex. This mode is default and works best in most cases.

  Disabled
  : Disables Sticky Selection. When you move a UV in this mode, each face owns its own UVs, allowing them to be separated.

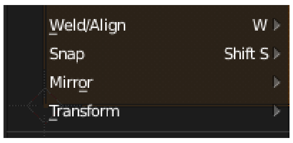
When Sync Selectionis **On** the following can be selected:

- Vertex

- Edge
- Face

  In this Mode, selection behaves differently. When selecting UVs or Edges, it behave like Shared Vertex mode above. When selecting Faces, it behaves as in Disabled Stick Selection above.

Transforming UVs



UV Transformation Menu.

UVs can be:

- Translated G
- Rotated R
- Scaled S

They can also be hidden or shown using the H and AltH respectively, the same way as in Edit Mode.

Axis Locking

Transformations can be locked to an axis by pressing X or Y after one of the transform tools. Also, holding the MMB will constrain movement to the X or Y axis.

Pivot Points

The UV editor has a 2D cursor. Its position can be changed by LMB clicking in the UV editor. You can also manually adjust its position in the Properties Panel. The range by default is from 0 to 256 starting from the lower left corner. By enabling Normalized under Coordinates, the range changes from 0 to 1.

The 2D Cursor can be snapped to nearest pixels or to selected elements, by selecting UVs Menu under Snap.

The Pivot Point can be changed to:

- Bounding Box Center
- Median Point
- 2D Cursor Location

Proportional Editing

Proportional Editing is available in UV editing. The controls are the same as in the 3D view. See Proportional Editing in 3D for full reference.

Snapping

Snapping in UV is also similar to Snapping in 3D, but only snapping to UVs works, however, the Snap to Pixels option in the UVs Menu will force the UVs to snap to the pixels of an image if loaded.

Additional tools can be found in the UVs Menu under the Snap Submenu:

Snap Pixels
  Moves selection to nearest pixel
Snap to Cursor
  Moves selection to 2D cursor location
Snap to Adjacent Unselected
  Moves selection to adjacent unselected element

Weld and Align

the Weld tool, W1 will move selected UVs to their average position

Align, W2,W3, and W4will line up selected UVs on the X axis, Y axis, or automatically chosen axis.

Mirror

Components can be mirrored on the Y axis or the X axis. You can select Mirror X and Mirror Y from the Snap sub menu in the UV menu.

You can also use the hotkey CtrlM then enter X or Y, or hold the MMB and drag in the mirror direction.

Stitch

Stitch, V, will join selected UVs that share Vertices. You set the tool to limit stitching by distance in the Tool Settings, by activating Use Limit and adjusting the Limit Distance

Minimize Stretch

the Minimize Stretch tool, CtrlV Reduces UV stretch by minimizing angles. This essentially relaxes the UVs

Reverse and Rotate UVs

Recall how the orientation of the UV Texture is relative to each face? Well, you might find that, for example, the image is upside down or laying on its side. If so, use Face->Rotate UVs (in the 3D window in Face Select mode) menu to rotate the UVs per face in 90-degree turns.

The Face->Reverse UVs to flips the image over like a pankcake in a pan, mirroring the UVs per face and showing you the image 'reversed'.

Pinning

When Unwrapping a model it is sometimes useful to "Lock" certain UVs, so that parts of a UV layout stay the same shape, and/or in the same place.

Pinning is done selecting a UV, then by selecting Pin from the UVs menu, or the shortcut P. You can Unpin a UV with the shorctut AltP

Pinning is most effective when using the Unwrap method of UV mapping, for organic objects. An example is when you are modeling a symmetrical object using the Mirror Modifier. Some of the UVs on the mirror axis may be shared across the mirrored counterparts. You could pin the UVs that correspond to the midline, then align them on the X axis, and they will stay in that location.

Pinning also work great with the Live Unwrap tool. If you pin two or more UVs, with Live Unwrap on, dragging pinned UVs will interactively unwrap the model. This helps with fitting a UV island to a certain shape or region.

Optimizing the UV Layout

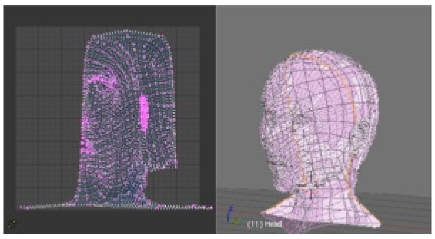
When you have unwrapped, possibly using seams, your UV layout may be quite disorganized and chaotic. You may need to proceed with the following tasks: Orientation of the UV mapping, arranging the UV maps, stitching several maps together.

The next step is to work with the UV layouts that you have created through the unwrap process. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you. In this fashion, you can use the UV Texture image to guide additional geometry changes.

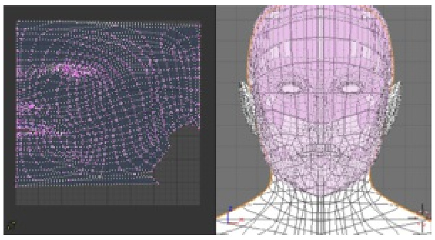
When arranging, keep in mind that the entire window is your workspace, but only the UV coordinates within the grid are mapped to the image. So, you can put pieces off to the side while you arrange them. Also, each UV unwrap is its own linked set of coordinates.

You can lay them on top of one another, and they will onionskin (the bottom one will show through the top one). To grab only one though, RMB select one of the UV coordinates, and use Select->Linked UVs (CtrlL) to select connected UVs, not border select because UVs from both will be selected.

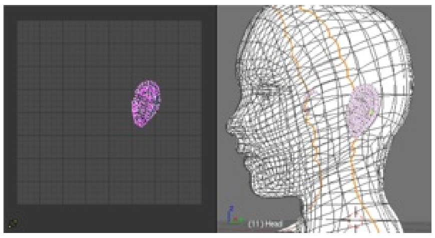
Combining UV Maps


Bad Unwrap-Note Ear and Neck

Very often you will unwrap an object, such as the face example we have been using, and get it 'mostly right' but with parts of the mesh that did not unwrap properly, or are horribly confusing. The picture to the right shows an initial unwrap of the face using the Unwrap from sphere option. The issues are with the ear; it is just a mush of UVs, and the neck, it is stretched and folded under. Too much work to clean up.


Unwrap Face Only, without Ear or Neck

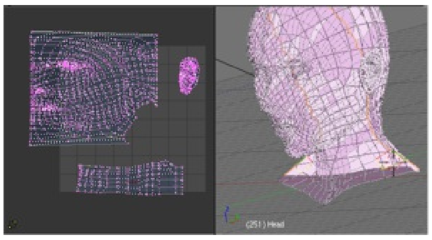
We can tell that the ear would unwrap nicely with just a straightforward projection from the side view, and the neck with a tubular unwrap. So, our general approach will be to unwrap different parts of the object (face, ears, and so on) using different unwrap calculations, selecting each calculation according to whatever works best for that piece. So let's begin: We select only the "face" faces, unwrap them using the *Sphere* calculation, and scale and rotate them somewhat to fit logically within the image area of the UV/Image Editor window pan.


Unwrap Projection: Ear

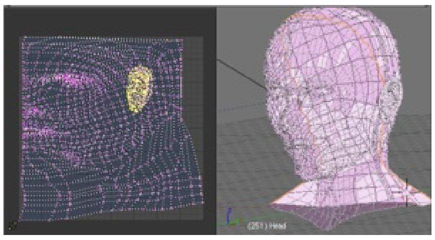
Once we're satisfied with the face, it's time to turn our attention to the ear. First, unselect the faces you were working with. Their UVs will disappear from the UV/Image Editor, but they are still there, just not shown. (To verify this, you can select a few faces in 3D view and it will show up in the UV/Image Editor.)

To work on the ear, in the 3D View, we now select only the "ear" faces. You can use Vertex Groups to select the ear faces. Selecting sub-meshes is easy too, since they are not connected to the rest of the mesh. Simply selecting Linked vertices will select that entire submesh. Basically, since you are in edit mode, all of the selecting/unselecting features are available to you.

Now re-unwrap the ear using the *Project* calculation from side view, and scale and rotate them somewhat (discussed in the next section), and place them off to the side. You can do this repetitively, using different UV calculations; each re-calculation just puts those UVs for the selected faces somewhere else. Choose the calculation for each piece that gives you the best fit and most logical layout for subsequent painting of that piece.


UV Maps together

When all of the pieces of the mesh have been unwrapped using the various calculations, you should end up with something that looks like to the Example to the right. All of the sections of the mesh have been mapped, and all those maps are laid out in the same UV Texture map. Congratulations! From here, it is a simple matter of "stitching" (discussed in the next section) to construct the entire UV Map as a single map.


UV Maps Arranged and Stitched

When you have completed arranging and stitching, you will end up with a consolidated UV Map, like that shown to the right, arranged such that a single image will cover, or paint, all of the mesh that needs detailed painting. All of the detailed instructions on how to do this are contained in the next section. The point of this paragraph is to show you the ultimate goal. Note that the mesh shown is Mirrored along the Z axis, so the right side of the face is virtual; it is an exact copy of the right, so only one set of UVs actually exist. (If more realism is desired, the *Mirror* modifier would be applied, resulting in a physical mirror and a complete head. You could then make both side physically different by editing one side and not the other. Unwrapping would produce a full set of UVs (for each side) and painting could thus be different for each side of the face, which is more realistic.)

Average Island Scale

Using the Average Island Scale tool, shortcut CtrlA, will scale each UV island so that they are all approximately the same scale.

Packing Islands

The Pack Islands tool, shortcut CtrlP, will uniformly scale, then individually transform each Island so that they fill up the UV space as much as possible. This is an important tool for efficiently making use of the texture space.

Constraining to Image Bounds

Turning on Constrain to Image Bounds will prevent UVs from being moved outside the 0 to 1 UV range.

Iteration and Refinement

At least for common people, we just don't "get it right the first time." It takes building on an idea and iterating our creative process until we reach that magical milestone called "Done." In software development, this is called the Spiral Methodology.

Applied to Computer Graphics, we cycle between modeling, texturing, animating, and then back to making some modifications to mesh, re-UV mapping, tweaking the animation, adding a bone or two, finding out we need a few more faces, so back to modeling, etc. We continue going round and round like this until we either run out of time, money, or patience, or, in some rare cases, are actually happy with our results.

Refining the Layout

Refinement comes into play when we finally look at our character, and realize that we need more detail in a particular spot. For example, areas around the eyes might need crow's feet, or we need to add a logo to the vest. As you start to edit the image, you realize that there just aren't enough pixels available to paint the detail that you want.

Your only choice is to expand the size (scale out) that UV face. Using the minimize stretch or scale commands, you expand the UV faces around the eyes or chest, allocating more pixels to those areas, but at the same time taking away pixels (detail) from something else, like the back of the head. After refining the UV map, you then edit the image so that it looks right and contains the details you want.

Reusing Textures

Another consideration is the need to conserve resources. Each image file is loaded in memory. If you can re-use the same image on different meshes, it saves memory. So, for example, you might want to have a generic 'face' painting, and use that on different characters, but alter the UV map and shape and props (sunglasses) to differentiate.

You might want to have a "faded blue jeans" texture, and unwrap just the legs of characters to use that image. It would be good to have a generic skin image, and use that for character's hands, feet, arms, legs, and neck. When modeling a fantasy sword, a small image for a piece of the sword blade would suffice, and you would Reset Unwrap the sword faces to re-use that image down the length of the blade.

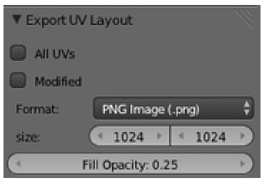
Applying Textures

Sooner or later, you may want to use an image texture on your model. If you are using an external application, you need to know where on the mesh you are painting. You may also need to test your UV mapping with a test image. This section covers how to export an outline of your UV map, and how to load images into the UV editor.

Exporting UV Layout Image

As a way of communicating to an artist who is painting your UV Texture for you, Blender has a tool called Save UV Face Layout (located in the UV/Image Editor Window, UVs->Save UV Face Layout) that saves an image as a Targa (.tga), EPS, or an SVG format for the object you have selected.

The image is an outline of the UV face mapping. Activating the tool brings up the File Browser Window with options for saving the layout:


▼ Export UV Layout
All UVs
Modified
Format: PNG Image (.png)
size: 1024 1024
Fill Opacity: 0.25

Export options

All UVs
: if disabled, then only the UV faces selected will be outlined

Modified
: Export UVs from the modified mesh.

Format
: Select the type of image file to save (.png, .eps, .svg)

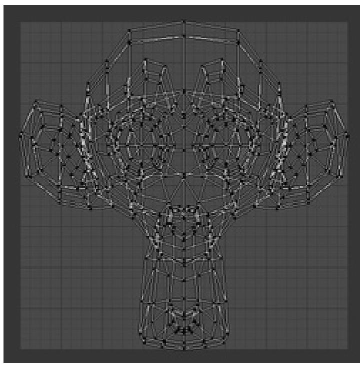
Size
: select the size of the image in pixels. The image be square.

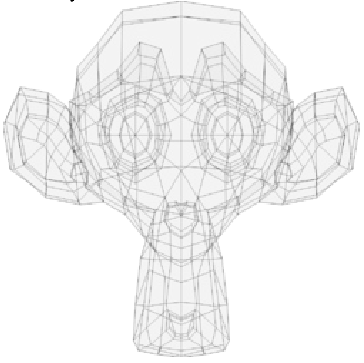
Fill Opacity
: Set the opacity of the fill

The image will be lines defining the UV edges that are within the image area of the UV mapping area. Edges outside the boundary, even if selected, will not be shown in the saved graphic.

The artist will use this as a transparent layer in their paint program as a guide when painting your texture. The example below shows Blender in the background, and the Gimp working on the texture, using the saved layout as a guide. Note that targa format supports the Alpha channel, so you can paint transparent areas of the mesh.

For using images as textures, see the page on Image Textures


A uv layout in the uv editor


A snapshot of the uv layout to be used in an image editor

Applying Textures to UVs

The UV/Image Editor allows you to map textures directly to the mesh faces. The 3D View window shows you the object being textured. If you set this window into Textured viewport shading, you will immediately see any changes made in the UV/Image Editor window in this window, and vice versa.

You can edit and load images, and even play a game in the Blender Game Engine with UV textures for characters and object, without a material, and still see them in the 3D window. This is because no 'real' rendering is taking place; it is all just viewport shading. If you were to apply an image to UVs then render, the texture would not show up by default

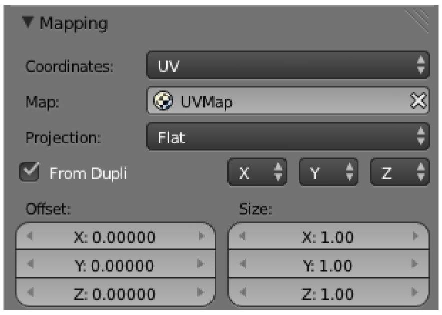
To render an image however, you must

1. create a Material for the object, and
2. tell Blender to use the UV Textures on faces when rendering.

To create a Material, you have to click Add New Material in the Shading context.

There are two ways to tell Blender to use the UV Texture when rendering: the Proper way and the Quick Way:

Use UV Coordinates


Mapping
Coordinates: UV
Map: UVMap
Projection: Flat
From Dupli ✓ X Y Z
Offset: X: 0.00000 Y: 0.00000 Z: 0.00000
Size: X: 1.00 Y: 1.00 Z: 1.00

A texture setup to map using its UV coordinates

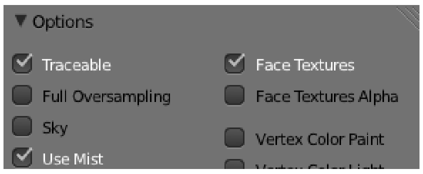
In the Texture channel panel, Add a New Texture and define the texture as an image and load the image you want to use. In the Mapping section, choose UV from the Coordinates menu, and select the UV layer to use.

Make sure it is mapped to Color in the Influence section as well (it will be mapped to Color by default, and the UV Texture is named "UVTex" by default). If the image has an alpha channel and you want to use it, click "UseAlpha" in the Map Image panel.

Full details of using Image textures are on the Image Textures page.

Material is Required for Rendering
You can perform UV Texturing on a mesh within Blender without assigning a material, and you will even see it in your 3D View in textured viewport mode. However, when you render, you will just get a default gray if the object does not have a Material assigned. You will get a black if you do not load an image. If you do not create a texture that uses the image, or enable Face Texture, your object will render according to the procedural material settings.

Face Textures


Options
✓ Traceable
Full Oversampling
Sky
✓ Use Mist
✓ Face Textures
Face Textures Alpha
Vertex Color Paint

The Material panel with activated Face Textures button.

An alternate way is to set up a Face Textures Material as shown. To do so, with the buttons window displayed, press F5 to display the Shader Buttons. In the Buttons window, Material settings, click ADD NEW material.

On the Options panel, enable Face Textures. This way is quick, but bypasses the normal rendering system for fast results, but results which do not respect transparency and proper shading.

Loading and Saving Images

In the UV editor, you can assign certain faces certain textures. To do so, first you need an image to work with. In the Image Menu you can open an image file with the File Browser. If you have images in the file already, that you want to use, click the Browse button in the Header, or make a new texture by clicking the New button.

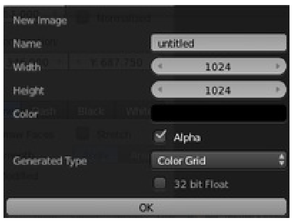
In a team environment, or if you are using an external paint program to edit the image while the .blend file is active, and the file is updated and re-saved, use the UV/Image Editor to Image->Reload it and see the latest and greatest in Blender. Also, use Reload if you have mapped more faces to an image, and the 3D View will be updated with the latest image mapping back to faces.

If you move the image file, Blender may not be able to find it, and you will have to Image->Replace it. Use this option to map a UV layout to a different image altogether.

Replacing the active Image

Recall that each face gets coordinates and a link to an image. To map a face to a different image, simply select that face (or faces) and use the UV/Image Editor window Image}} menu to Replace the current image with an existing file (such as a JPG or PNG file).

New Images


New Image
Name: untitled
Width: 1024
Height: 1024
Color
☑ Alpha
Generated Type: Color Grid
☐ 32 bit Float
OK

The new Image dialogue

When you select New Image you are presented with several options. This Generated image can also be modified afterward in the Properties Panel:

Image Name
: Set the name if the generated image

Width and Height
: Set the size if the image in pixels

Color
: Sets the the default fill color if creating a blank image.

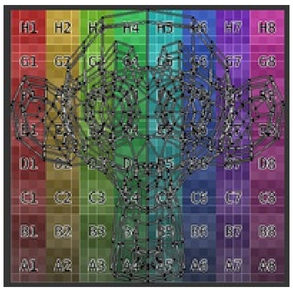
Alpha
: Adds an alpha channel to the image

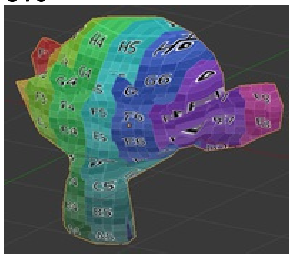
Generated Type
: The type of image to generate:
: UV Grid
: Creates a checkerboard pattern with a colored + in each square.
: Color Grid
: Creates a UV Test Grid, which is useful for testing how UVs have been mapped, and to reduce stretching. There are two types available, which can be set after the image has been created.
: Blank
: Generates a blank image of the specified color.

32 bit
: Creates a 32 bit image. This is a larger file size, but holds much more color information than the standard 8 bit image. For close ups and large gradients, it may be better to use a 32 bit image.

Using the Test Grid

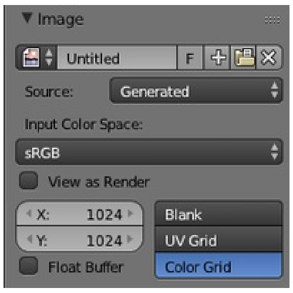
Use the UV Test Grid option to check for undue stretching or distortion of faces. If your image is a base uniform pattern and you want the application of that image to your model to look like cloth, you do NOT want any stretching (unless you want the cloth to look like spandex).


The test grid applied to the UVs


A preview of the texture on the geometry

When you render, the mesh will have the test grid as its colors, and the UV Texture will be the size image you specified. You can save the UV image using the Image->Save menu.

Image Settings



«Image» section on «Properties» panel in UV/Image Editor

When an image has been loaded or created in the UV editor, an additional section appears in the Properties Panel. The first row of buttons allow you to:

- Browse for an image
- Change the image name
- Set as Fake User
- Create a New Image
- Open an image
- Unlink Datablock

Select the image type in the Source menu. Each has different options:

Generated
: Generates a new image:

  Width and Height of image in pixels
  Blank
  : Creates a Blank image

  UV grid
  : Creates a checkerboard pattern with colored plus symbols in each square.

  Color Grid
  : Creates a more complex colored grid with letters and numbers denoting locations in the grid.

File
: Use for loading image files:

  Fields
  : Use if image is made of fields. You can set it to use Upper First or Lower First

  Premultiply
  : Converts RGB from key alpha to premultiplied alpha.

Movie and Sequence
: Frames
  : Set the number of frames to use

  Start
  : Set the starting frame of the movie/sequence

  Offset
  : Offset the number of frame used in the animation

  Fields
  : Set the number fields per rendered frame to use(2 fields is 1 frame)

  Auto Refresh
  : Always refresh images on frame changes.

Cyclic

Cycle the images in a movie/sequence.

Saving Images

Images can be saved to external files if they were created or edited in Blender with tools in the Image menu. If images are already files, use the Save command (AltS). You can also Save As (F3) if the image was generated or you want to save as a different name. Using Save as Copy, (F3) will save the file to a specified name, but will keep the old one open in the Image editor.

Modifying your Image Texture

To modify your new Texture, you can:

- Render Bake an image based on how the mesh looks
  - The Render Bake feature provides several tools to replace the current image based on a render of Vertex Paint colors, Normals (bumps), Procedural materials, textures and lighting, and ambient occlusion.
- Paint using Texture Paint.
  - Use the UV/Image Editor menu *Image->New*. Then start painting your mesh with
- Use external software to create an image
  - Using your favorite image painting program, you could use an exported UV layout to create a texture. Then save your changes, and back in Blender, use the Image->Open menu command to load it as your UV image for the mesh in Face Select Mode for the desired (and active) UV Texture layer. Using the Edit Externally tool in the Image menu, Blender will open an image editor, as specified in the User Preferences and load in the image to be edited.
- Use the "projection painting" feature of recent versions of Blender
- Use the Bake uV-Textures to Vertex Colors addon to create an image from vertex colors
- Some combination of the above.

The first three options, (UV Painter, Render Bake, and Texture Baker) replace the image with an image that they create. Texture paint and external software can be used to add or enhance the image. Regardless of which method you use, ultimately you must either

- save your texture in a separate image file (for example JPG for colors, PNG with RGBA for alpha),
- pack the image inside the blend file (UV/Image Editor Image->Pack as PNG),
- or do both.

The advantage to saving as a separate file is that you can easily switch textures just by copying other image files over it, and you can use external editing programs to work on it. The advantage of packing is that your whole project is kept in the .blend file, and that you only have to manage one file.

You can invert the colors of an image by selecting the Invert menu. in the Image menu

Packing Images inside the Blend file

If you pack your .blend file, the current version of all UV Texture images are packed into the file. If those files later change, the updates will not be automatically re-packed; the old version of the image is what will be used. To update, you will have to re-pack or reload.

To pack an image, select Pack Image from the Image menu. To Unpack, select this option again and select Remove Pack.

The File->Append function automatically goes into .blend files and shows you the image textures packed in it. The public domain Blender Texture CD is also a great resource, and there are many other sources of public domain (and licensed) textures. All textures on the Elephants Dream CD are liberally licensed under CC-BY 2.5.

Page status (reviewing guidelines)

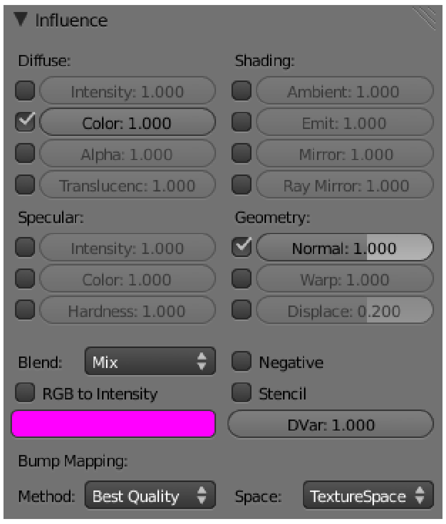
Partial page Text elaborate
Proposed fixes: none

Material Textures Influence

Not only can textures affect the color of a material, they can also affect many of the other properties of a material. The different aspects of a material that a texture influences are controlled in the Influence panel.

Note
Texture options for Surface and Wire materials and in some cases also for Volume and Halo materials.

Surface and Wire materials


Influence
Diffuse:
Intensity: 1.000
[✓] Color: 1.000
Alpha: 1.000
Translucenc: 1.000
Specular:
Intensity: 1.000
Color: 1.000
Hardness: 1.000
Shading:
Ambient: 1.000
Emit: 1.000
Mirror: 1.000
Ray Mirror: 1.000
Geometry:
[✓] Normal: 1.000
Warp: 1.000
Displace: 0.200
Blend: Mix
Negative
RGB to Intensity
Stencil
DVar: 1.000
Bump Mapping:
Method: Best Quality
Space: TextureSpace

Texture Influence panel for a Surface material

Diffuse

Intensity
: Amount texture affects affects diffuse reflectivity

Color
: Amount texture affect the basic color or RGB value of the material

Alpha
: Influences the opacity of the material. See Use Alpha for Object Transparency. Also use Z Transparency for light and if combining multiple channels.

Translucency
: Influences the Translucency amount.

Specular

Intensity
: Amount texture affect specular reflectivity

Color
: Influences the Specular color, the color of the reflections created by the lamps on a glossy material.

Hardness
: Influences the specular hardness amount. A DVar of 1 is equivalent to a Hardness of 130, a DVar of 0.5 is equivalent to a Hardness of 65.

Shading

Ambient
: Influences the amount of Ambient light the material receives.

Emit
: Influences the amount of light Emitted by the material.

Mirror
: Influences the mirror color. This works with environment maps and raytraced reflection.

Ray Mirror
: Influences the strength of raytraced mirror reflection.

Geometry

Normal
: Commonly called bump mapping, this alters the direction of the surface normal. This is used to fake surface imperfections or unevenness via bump mapping, or to create reliefs.

Warp
: Warp allows textures to influence/distort the texture coordinates of a next texture channel. The distortion remains active over all subsequent channels, until a new Warp has been set. Setting the factor at zero cancels out the effect.

Displace
: Influences the Displacement of vertices, for using Displacement Maps.

Other Controls

Blend
: Blending operation to perform. See Texture Blending Modes for details.

RGB to intensity
: With this option enabled, an RGB texture (affects color) is used as an intensity texture (affects a value).

Blend Color
: If the texture is mapped to Col, what color is blended in according to the intensity of the texture? Click on the swatch or set the RGB sliders.

Negative
: The effect of the Texture is negated. Normally white means on, black means off, Negative reverses that.

Stencil
: The active texture is used as a mask for all following textures. This is useful for semitransparent textures and "Dirt Maps". Black sets the pixel to "untexturable". The Stencil mode works similar to a layer mask in a 2D program. The effect of a stencil texture can not be overridden, only extended. You need an intensity map as input.

DVar
: Destination Value (not for RGB). The value with which the Intensity texture blends with the current value. Two examples:

- The Emit value is normally 0. With a texture mapped to Emit you will get maximal effect, because DVar is 1 by default. If you set DVar to 0 no texture will have any effect.

- If you want transparent material, and use a texture mapped to Alpha, nothing happens with the default settings, because the Alpha value in the Material panel is 1. So you have to set DVar to 0 to get transparent material (and of course Z Transparency also). This is a common problem for beginners. Or do it the other way round - set Alpha to 0 and leave Dvar on 1. Of course the texture is used inverted then.

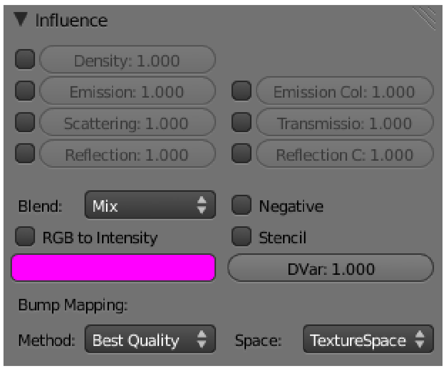
Bump Mapping
: Settings for bump mapping.

  Method

  Best Quality, Default, Compatible, Original

  Space
  : Texture Space, Object Space, View Space

Volume materials



Texture Influence panel for Volume material

Special texture options for Volume materials

Density
: Causes the texture to affect the volume's density.

Emission
: Causes the texture to affect the volume's emission.

Scattering
: Amount the texture affects scattering.

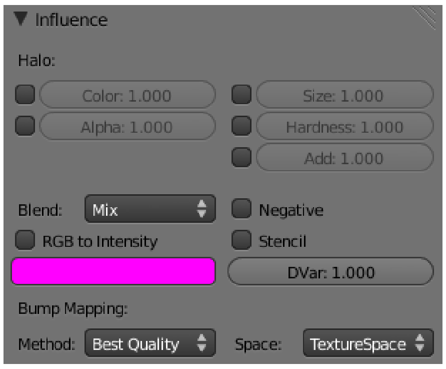
Reflection
: Amount the texture affects brightness of out-scattered light

Emission Color
: Amount the texture affects emission color.

Transmission
: Amount the texture affects result color after light has been scattered/absorbed.

Reflection Color
: Amount the texture affects color of out-scattered light.

Halo materials



Texture Influence panel for a Halo material

Special texture options for Halo materials

Size
: Amount the texture affects ray mirror.

Hardness
: Amount the texture affects hardness.

Add
: Amount the texture affects translucency.

Texture Blending Modes

Blending Modes are different methods of controlling how the texture influences material properties. While a blending mode defines the specific operation performed, blending factor controls the amount, the overall "strength" of this operation. For textures such blending factor is set via sliders in the Influence panel. Throughout this section, the term base layer refers to the base material color being manipulated (as defined by texture's Influence) and blend layer refers to the texture. Following is a list of available texture blending modes:

Linear Light

Brightens base layer depending on blend layer. If blend layer is more than 50% bright, base layer is brightened by the blend layer values, otherwise it is darkened by the blend layer values.

Soft Light

Lightens or darkens base layer depending on the blend layer brightness. The effect is softer than that of Linear Light or Overlay modes, with pure white and pure black blend layers not yielding pure white/black results.

Color

Mixes hue and saturation of the blend layer into the base layer.

Value

Mixes value of the blend layer into the base layer.

Saturation

Mixes saturation of the blend layer into the base layer.

Hue

Mixes hue of the blend layer into the base layer.

Overlay

Combines Multiply and Screen modes. The darker the base layer, the more the blend layer influences the mix.

Lighten

Any fragments of the base layer that are darker than those of the blend layer are replaced by the blend layer fragments.

Darken

Any fragments of the base layer that are lighter than those of the blend layer are replaced by the blend layer fragments.

Difference

Mixes with absolute value of the difference between base and blend layers.

Divide

Base layer is divided by the blend layer.

Screen

Inverse of the base layer is multiplied by the blend layer.

Subtract

Blend layer is subtracted from the base layer.

Multiply

Base layer is multiplied by the blend layer.

Add

Blend layer is added to the base layer.

Mix

Linear interpolation between base and blend layers.

Bump and Normal Maps

Description

Normal Maps and Bump Maps both serve the same purpose: they simulate the impression of a detailed 3D surface, by modifying the shading as if the surface had lots of small angles, rather than being completely flat. Because it's just modifying the shading of each pixel, this will not cast any shadows and will not obstruct other objects. If the camera angle is too flat to the surface, you will notice that the surface is not really shaped.

Both Bump Maps and Normal Maps work by modifying the normal angle (the direction pointing perpendicular from a face), which influences how a pixel is shaded. Although the terms Normal Map and Bump Map are often used synonymously, there are certain differences.

Bump maps
: These are textures that store an **intensity**, the relative height of pixels from the viewpoint of the camera. The pixels seem to be moved by the required distance in the direction of the face normals. (The "bump" consists only of a displacement, which takes place along the existing, and unchanged, normal-vector of the face.) You may either use greyscale pictures or the intensity values of a RGB-Texture (including images).

Normal maps
: These are images that store a **direction**, the direction of normals directly in the RGB values of an image. They are much more accurate, as rather than only simulating the pixel being away from the face along a line, they can simulate that pixel being moved at any direction, in an arbitrary way. The drawbacks to normal maps are that unlike bump maps, which can easily be painted by hand, normal maps usually have to be generated in some way, often from higher resolution geometry than the geometry you're applying the map to.

Normal maps in Blender store a normal as follows:

- Red maps from (0-255) to X (-1.0 - 1.0)
- Green maps from (0-255) to Y (-1.0 - 1.0)
- Blue maps from (0-255) to Z (0.0 - 1.0)

Since normals all point towards a viewer, negative Z-values are not stored (they would be invisible anyway). In Blender we store a full blue range, although some other implementations also map blue colors (128-255) to (0.0 - 1.0). The latter convention is used in "Doom 3" for example.

Workflow

The steps involved in making and using Bump and Normal Maps is:

1. Model a highly detailed ("hi-poly") model
2. Bake the Bump and/or Normal maps
3. Make a low-poly, less detailed model
4. Map the map to the low-poly model using a common coordinate system

Consult the Modeling section for how to model a highly detailed model using the Mesh tools. How much detail you put in is totally up to you. The more ridges and details (knobs, creases, protrusions) you put in, the more detailed your map will be.

Baking a map, simply put, is to take the detail of a high polygon mesh, and apply it to a similar object. The similar object is identical to the high-poly mesh except with less vertices. Use the Render Bake feature in Blender to accomplish this.

Modeling a low-poly using Blender's Mesh editing tools. In general, the same or similar faces should exist that reflect the model. For example, a highly detailed ear may have 1000 faces in the high-poly model. In the low-poly model, this may be replaced with a single plane, oriented in the same direction as the detailed ear mesh. *(Tip:* Blender's multi-resolution mesh modeling feature can be used to good effect here.)

Mapping is the process of applying a texture to the low-poly mesh. Consult the Textures Mapping section for more information on applying a texture to a mesh's material. Special considerations for Bump and Normal Maps is:

- When using a Bump map, map the texture to Normal and enable No RGB.
- When using a Normal map, map the texture to Normal.

The coordinate systems of the two objects must match. For example, if you bake using a UV map of the high-poly model, you must UV map the low poly model and line up its UV coordinates to match the outline of the high-poly image (see UV unwrapping to line up with the high-poly map edges.

Displacement Maps

Description

Displacement mapping allows a texture input to manipulate the position of vertices on rendered geometry. Unlike Normal or Bump mapping, where the shading is distorted to give an illusion of a bump (discussed on the previous page), Displacement Maps create real bumps, creases, ridges, etc in the actual mesh. Thus, the mesh deformations can cast shadows, occlude other objects, and do everything that changes in real geometry can do, but, on the other hand, requires a lot more vertices to work.

Options

In the Influence panel, the strength of the displacement is controlled by the Displace and Normal sliders.

- If a texture provides only normal information (e.g. Stucci), vertices move according to the texture's normal data. The normal displacement is controlled by the Normal slider.
- If a texture provides only intensity information (e.g. Magic, derived from color), vertices move along the directions of their normals (a vertex has no normal itself, it's the resulting vector of the adjacent faces). White pixels move outward in the direction of the normal, black pixels move in the opposite direction. The amount of displacement is controlled with the Displace slider.

The two modes are not exclusive. Many texture types provide both information (Clouds, Wood, Marble, Image). The amount of each type can be mixed using the respective sliders. Intensity displacement gives a smoother, more continuous surface, since the vertices are displaced only outward. Normal displacement gives a more aggregated surface, since the vertices are displaced in multiple directions.

The depth of the displacement is scaled with an object's scale, but not with the relative size of the data. This means if you double the size of an object in object mode, the depth of the displacement is also doubled, so the relative displacement appears the same. If you scale inside Edit Mode, the displacement depth is not changed, and thus the relative depth appears smaller.

Hints

Displacement maps move the rendered faces, not the physical mesh faces. So, in 3D View the surface may appear smooth, but render bumpy. To give a detailed surface, there has to be faces to displace and have to be very small. This creates the trade-off between using memory and CPU time versus render quality.

From best to worst, displacement works with these object types using the methods listed to control the render face size:

Subdivision Surface Meshes
: Rendered face size is controlled with render subsurf level. Displacement really likes smooth normals.

Manually (Edit Mode) subdivided meshes
: Control render faces with number of subdivides. (This can be combined with the above methods.) Displaces exactly the same Simple Subsurf, but slows editing down because of the OpenGL overhead of drawing the extra faces. (You can't turn the edit subdivide level down this way).

Meta Objects
: Control render faces with render wiresize. Small wire == more faces.

The following are available, but currently don't work well. It is recommended that you convert these to meshes before rendering.

Open NURBS Surfaces
: Control render faces with U/V Surface Resolution. Higher numbers give more faces. (Note normal errors).

Closed NURBS Surfaces
: Control with Surface Resolution controls. (Note the normal errors, and how implicit seam shows).

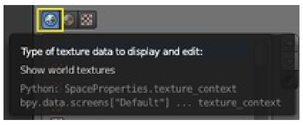
Curves and Text
: Control with Surface Resolution controls. Higher gives more render faces. (Note that the large flat surfaces have few render faces to displace).

Displace Modifier
If you want more control over your displacement, you'll probably want to use the Displace Modifier. This feature has lots of different options so that you can customize the displacement exactly to your liking.

World

Instead of a color, or blend of two colors, Blender can use an 2D image which it maps to a very large Box or sphere which encompasses the entire scene, or which it maps to a virtual space around the scene.

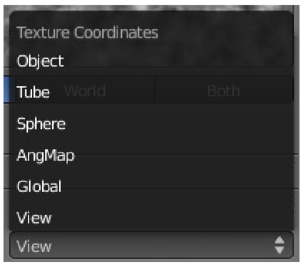
Options


Type of texture data to display and edit:
Show world textures
Python: SpaceProperties.texture_context
bpy.data.screens["Default"] ... texture_context

World textures button

The World textures are accessible in the texture menu (just select World first, then Texture).

Texture coordinates



Texture Coordinates popup menu

The World textures are used much like the Materials textures, except for a couple of differences. The textures can be mapped according to:

View
: The default orientation, aligned with the co-ordinates of the final render

Global
: Uses global coordinates

AngMap
: Used to wrap a standard hemisphere angular map around the scene in a dome. This can be used for image based lighting with Ambient Occlusion set to sky color. You'll generally need a high dynamic range image (HDRI) angular map. (It will look like a weird spherical image).

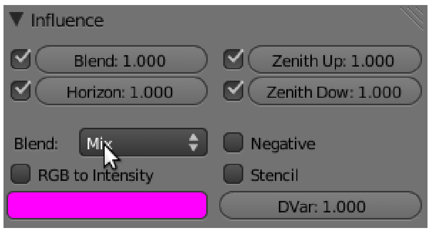
Sphere
: Sphere mapping, similar to that of materials

Tube
: Wrap the rectangular texture around in a cylinder, similar to that of materials

Object
: Position the texture relative to a specified object's local texture space

Equirectangular
: For 360 degree panorama sky, equirectangular mapping

Influence


Influence
✓ Blend: 1.000 ✓ Zenith Up: 1.000
✓ Horizon: 1.000 ✓ Zenith Dow: 1.000
Blend: Mix Negative
RGB to Intensity Stencil
DVar: 1.000

Texture Influence panel

The texture affects color only, but in four different ways:

Blend
: Makes the Horizon color appear where the texture is non-zero

Horizon
: Affect the color of the horizon

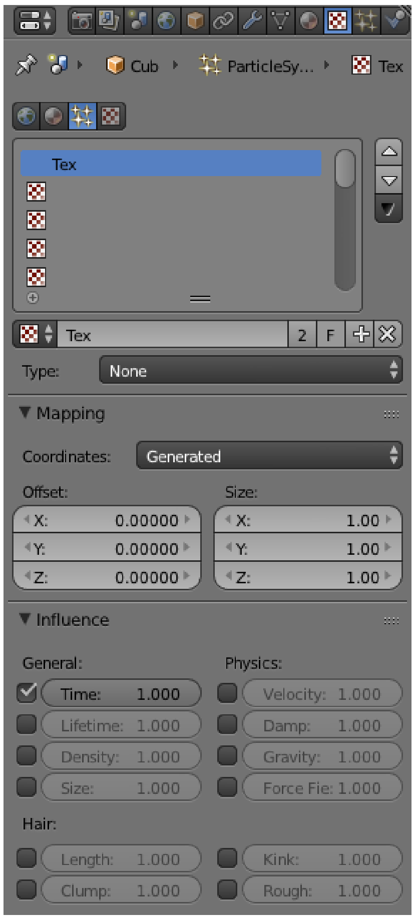
Zenith Up
: Affect the zenith color overhead

Zenith Down
: Affect the zenith color underneath

If you are disappointed that your camera appears to carry the texture with it rather than rotate through the texture, you should check the Real Sky checkbox in the World tab of the Properties view.

Particles

Blender allows to mapping textures on a Particles systems after they had been created.

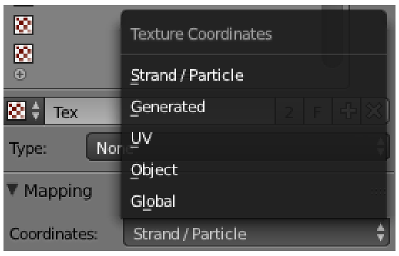

Tex
Type: None
Mapping
Coordinates: Generated
Offset: X: 0.00000 Y: 0.00000 Z: 0.00000
Size: X: 1.00 Y: 1.00 Z: 1.00
Influence
General: [✓ Time: 1.000] Lifetime: 1.000 Density: 1.000 Size: 1.000
Physics: Velocity: 1.000 Damp: 1.000 Gravity: 1.000 Force Fie: 1.000
Hair: Length: 1.000 Clump: 1.000 Kink: 1.000 Rough: 1.000

Common texture mapping settings for Particles systems

They are used much like the Materials textures, except for a couple of differences.

Options

Texture Coordinates



Texture coordinates for Particles systems

Global
: The scene's global 3D coordinates. This is also useful for animations; if you move the object, the texture moves across it. It can be useful for letting objects appear or disappear at a certain position in space.

Object
: Uses an object as source for coordinates. Often used with an Empty, this is an easy way to place a small image at a given point on the object (see the example below). This object can also be animated, to move a texture around or through a surface.

  Object
  : Select the name of an object.

Generated
: The original undeformed coordinates of the object. This is the default option for mapping textures.

UV
: UV mapping is a very precise way of mapping a 2D texture to a 3D surface. Each vertex of a mesh has its own UV co-ordinates which can be unwrapped and laid flat like a skin. You can almost think of UV coordinates as a mapping that works on a 2D plane with its own local coordinate system to the plane on which it is operating on. This mapping is especially useful when using 2D images as textures, as seen in UV Mapping. You can use multiple textures with one set of UV coordinates.

Strand/Particle
: Uses normalized 1D strand texture coordinate or particle age(X) and trail position (Y). Use when texture is applied to hair strands or particles.

Influence

General

Time
: Affect the emission time of the particles

  Lifetime
  : Affect the life time of the particles

  Density
  : Affect the density of the particles

  Size
  : Affect the particles size

Physics

Velocity
: Affect the particles initial velocity

  Damp
  : Affect the particles velocity damping

  Gravity
  : Affect the particles gravity

  Force Fiels
  : Affect the particles force fields

Hair

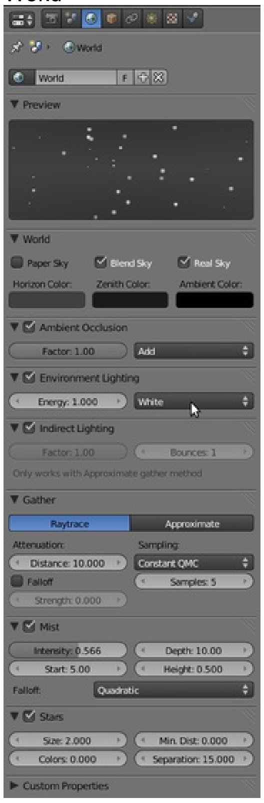
Length
: Affect the child hair length

  Clump
  : Affect the child clumping

  Kink
  : Affect the child kink

  Rough
  : Affect the child hough

World


World
World F
Preview
World
Paper Sky ☐　Blend Sky ✓　Real Sky ✓
Horizon Color:　Zenith Color:　Ambient Color:
Ambient Occlusion ✓
Factor: 1.00　Add
Environment Lighting ✓
Energy: 1.000　White
Indirect Lighting ✓
Factor: 1.00　Bounces: 1
Only works with Approximate gather method
Gather
[✓ Raytrace] Approximate
Attenuation:　Sampling:
Distance: 10.000　Constant QMC
Falloff ☐　Samples: 5
Strength: 0.000
Mist ✓
Intensity: 0.566　Depth: 10.00
Start: 5.00　Height: 0.500
Falloff: Quadratic
Stars ✓
Size: 2.000　Min. Dist: 0.000
Colors: 0.000　Separation: 15.000
Custom Properties

World panel

Blender provides a number of very interesting settings to complete your renderings by adding a nice background, and some interesting 'depth' effects. These are accessible via the World context. By default a very plain uniform world is present. You can edit it or add a new World.

You have:

Background
: The color and texture of the world background, with special settings for mapping coordinates.

Mist
: Add a mist to your scene to enhance the feeling of depth.

Stars
: Randomly covers the background with halo-like dots (support only to Blender version 2.69).

While these world settings offers a simple way of adding effects to a scene, compositing nodes are often preferred, though more complex to master, for the additional control and options they offer. For example, filtering the Z value (distance from camera) or normals (direction of surfaces) through compositing nodes can further increase the depth and spacial clarity of a scene.

Note
Some of the settings under the World panel in Blender affect lighting so you find them under the Lighting chapter (see Ambient Light, Exposure and Ambient Occlusion). When using a Sun Lamp options for Sky & Atmosphere are available in the Lamp menu.

World Background

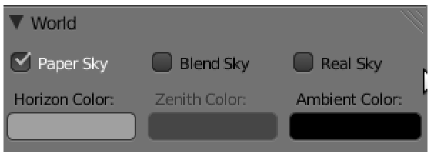
Description

The world buttons let you set up the shading of your scene in general. It can provide ambient color, and special effects such as mist, but a very common use of a World is to shade a background color.

Background Image in Render
To use an image as your render background, see BackBuf images specified in the Output Panel

Background Image in 3D
To use an image as a background image in your 3D view, for example as a reference when doing a model, see using a Background Image

Options



World panel

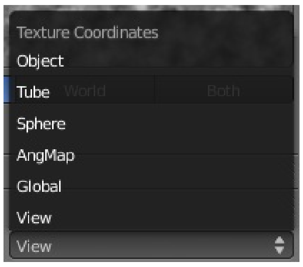
Horizon Color
    The RGB color at the horizon
Zenith Color
    The RGB color at the zenith (overhead)

How these colors are interpreted depends on which kind of Sky is chosen.

None Enabled
    If none of these three buttons is checked, your background will just be plain flat color (using the horizon one).
Paper Sky
    If this option is added, the gradient keeps its characteristics, but it is clipped in the image (it stays on a horizontal plane (parallel to x-y plane): what ever the angle of the camera may be, the horizon is always at the middle of the image).
Blend Sky
    The background color is blended from horizon to zenith. If only this button is pressed, the gradient runs from the bottom to the top of the rendered image regardless of the camera orientation.
Real Sky
    If this option is added, the gradient produced has two transitions, from nadir (same color as zenith) to horizon to zenith; the blending is also dependent on the camera orientation, which makes it more realistic. The horizon color is exactly at the horizon (on the x-y plane), and the zenith color is used for points vertically above and below the camera.

Textures

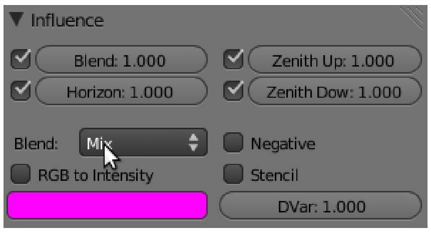
Instead of a color, or blend of two colors, Blender can use an 2D image which it maps to a very large Box or sphere which encompasses the entire scene, or which it maps to a virtual space around the scene.



Texture Coordinates popup menu

The World textures are accessible in the texture menu (just select World first, then Texture). They are used much like the Materials textures, except for a couple of differences. The textures can be mapped according to:

View
    The default orientation, aligned with the co-ordinates of the final render
Global
    Uses global coordinates
AngMap
    Used to wrap a standard hemisphere angular map around the scene in a dome. This can be used for image based lighting with Ambient Occlusion set to sky color. You'll generally need a high dynamic range image (HDRI) angular map. (It will look like a weird spherical image).
Sphere
    Sphere mapping, similar to that of materials
Tube
    Wrap the rectangular texture around in a cylinder, similar to that of materials
Object
    Position the texture relative to a specified object's local texture space



Texture Influence panel

The texture affects color only, but in four different ways:

Blend
: Makes the Horizon color appear where the texture is non-zero

Horizon
: Affect the color of the horizon

Zenith Up
: Affect the zenith color overhead

Zenith Down
: Affect the zenith color underneath

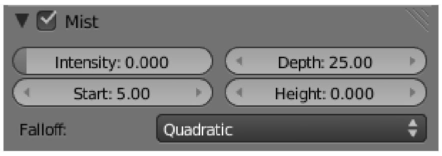
If you are disappointed that your camera appears to carry the texture with it rather than rotate through the texture, you should check the Real Sky checkbox in the World tab of the Properties view.

Mist

Description

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender makes objects farther away more transparent (decreasing their Alpha value) so that they mix more of the background color with the object color. With Mist enabled, the further the object is away from the camera the less it's alpha value will be.

Option


Mist
Intensity: 0.000
Depth: 25.00
Start: 5.00
Height: 0.000
Falloff: Quadratic

Mist panel

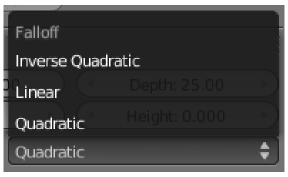
To enable mist in Blender
: Go to the scene properties and ensure "Mist" checked in under the "Passes" panel

Mist check box
: Toggles mist on and off

Minimum
: An overall minimum intensity, or strength, of the mist.

Start
: The distance from the camera at which the mist starts to fade in

Depth
: The distance from Start of the mist, that it fades in over. Objects further from the camera than Start+Depth are completely hidden by the mist.



Mist Falloff popup menu

Height
: Makes the mist intensity decrease with height, for a more realistic effect. If greater than 0, it sets, in Blender units, an interval around z=0 in which the mist goes from maximum intensity (below) to zero (above).

Falloff
: The decay rate of the mist (Quadratic/Linear/Inverse Quadratic). These settings control the rate of change of the mist's strength further and further into the distance.

Mist distances
To visualize the mist distances in the 3D View, select your camera, go to the camera menu, and enable Show Mist.

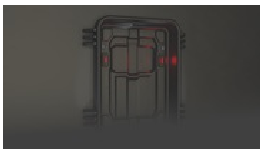
The camera will show mist limits as a line projecting from the camera starting from Start and of distance Depth.

To get a better view to evaluate the Mist visualization, ⇧ Shift1 NumPad with the camera selected (5 NumPad to toggle perspective view on and off). This will place the 3D view right over the camera looking down.

Transparency

Because Mist works by adjusting transparency, this can sometimes cause objects to be partially transparent when they shouldn't be. One workaround is to set the Mist settings as desired, but turn Mist off. The Mist data is still available for compositing even though it is off. Use Do Composite and the Nodes Editor to feed the Mist pass to an AlphaOver to blend the background color (or a render layer with just the sky) with the rendered image. This produces the mist effect but since Mist is off the object transparency (or lack of) is preserved.

Examples


Mist example

In this example (.blend) the Mist Height options has been limited to create smoke covering the floor.

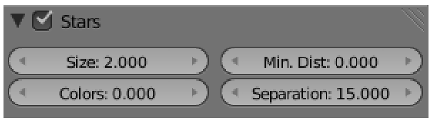
This simple scene was inspired by Stefan Morell's *Arc Sci-Fi Corridor*.

Stars

Description

Stars are randomly placed halo-like objects which appear in the background. Not for use with Cycles renderer.

Options



Star panel

Size
: The actual size of the star halo. It is better to keep it much smaller than the proposed default, to keep the material smaller than pixel-size and have pin-point stars. This is much more realistic.

Colors
: Adds a random hue to the otherwise plain white stars.

Min. Dist
: The *minimum* distance from the camera at which stars are placed. This should be greater than the distance from the camera to the *furthest* object in your scene, unless you want to risk having stars *in front* of your objects.

Separation
: The *average* distance between stars. Stars are intrinsically a 3D feature, they are placed in space, not on the image.

Introduction

After completing your character, you need to manipulate it for animation or just for posing. Rigging is the process of attaching a skeleton to your character mesh object so you can deform and pose it in different ways.

These actions do not fundamentally alter the mesh, and can easily be changed, undone, or combined with other poses.

note for editors
below we need to add more introductory text for all steps in rigging!

The following is a typical workflow for rigging:

1. Add an armature, which starts out with a single bone.
2. Add more bones as needed and link them together to form "limbs".
   (you can either extrude an existing bone, which parents it to the extruded bone automatically; or add new bones first and then link them together.)
3. Edit the bones to give proper proportions to the skeleton
4. Apply constraints to the joints
   (e.g. a human elbow can bend through about 170 degrees in one plane only)
5. Give a 'rest' (default) position to the skeleton.
6. Apply a mesh (body) to the armature (skinning)
7. Define how the movement of the armature affects the skin
   (folding, flexing, bulging)
8. Give poses to the armature.
   (There are multiple methods: By arranging each bone of the armature manually, or by copying a template armature, or by arranging the bones to follow a curve, or by making the armature follow externally collected motion-capture data.)
9. Check how the armature movement affects the skin, and adjust the parameters.
   (adjust the topology of the skin to make it look more natural)

All these steps are explained in details below.

-done! Please check+edit --Raindrops 19:32, 31 May 2013 (CEST)

Armatures

Armatures are like the real-life skeletons; and provide the structure for a mesh for the purpose of posing or animation.

Armature and Bone Panels
: shows how to use the different panels in Blender to adjust the armatures and bones.

Bones
: explains properties of Bones, which are the basic elements of armatures.

Visualization
: how to display bones in four different ways.

Structure
: Explains the structure of bones in an armature.

Selecting
: Select only the section of your armature that matters to you.

Editing

Bones
: Learn how to practically edit bones in Blender and see what that causes.

Sketching
: Use the Skeleton Sketching tool to easily sketch bones and bring them to reality in Blender.

Templating
: Templates offer a great way to quickly reuse already created rigs for your own models.

Skinning

This section shows how to "flesh out" your character from a given armature.

In normal English, "to skin" means 'to peel off skin', but here it is just the reverse (used in the sense of covering the armature with a skin): You will be putting a body (mesh) around an armature.

Linking Objects to Bones
: How to parent a bone to an object, so that the bone controls that object. This type of linking is used to simulate mechanical linkage (for example, Newton's cradle) or where the parts of the mesh are not deformed when the armature moves, as in case of modeling an insect body, crab, etc.

Skinning to Objects' Shapes
: How to attach the armature so that each of its bones controls a specific part of the “skin” object’s geometry. This type of linkage is used when the object surface flexes when the armature moves, such as bulging of biceps when the arm is folded.

Retargeting
: How to apply motion-capture data (acquired from real world) to a rig, so that it mimics the original movements realistically. This method also avoids laborious programming of each movement.

Posing

Posing means shaping and arranging the objects in your scene in a particular way to create an interesting composition. For example, look at the body language of The Thinker, or think of a scorpion raising its tail to strike.

Poses are also used to create animation. For example, to create animation of a tennis player serving a ball, you would have to create poses at different moments of the stroke: (a) when she holds the ball and racket at waist height (b) when she tosses the ball up, (c) when she strikes the ball, and (d) when her racket reaches at the lowest point after the strike (follow through). Then Blender creates all the intermediate poses to create the animation.

Visualization
: describes the visual aids that help you in posing the armature; especially for animation.

Editing Poses
: how to create a pose, and how to edit it to create the snapshots of an animation at different moments.

Pose Library
: storing frequently used poses or existing poses from another armature, so that they can be quickly accessed and applied.

Using Constraints
: how to apply constraints to bones so that they cannot form an unnatural pose.

Inverse Kinematics
: a feature where you move the last bone in a chain, and Blender automatically moves the whole chain accordingly. This is like lifting someone's finger: His whole hand automatically follows that movement.

Spline IK
: a feature where you can align a chain of bones along a curve.

Armatures

An "armature" is a type of object used for rigging. Armature object borrows many ideas from real life skeletons.

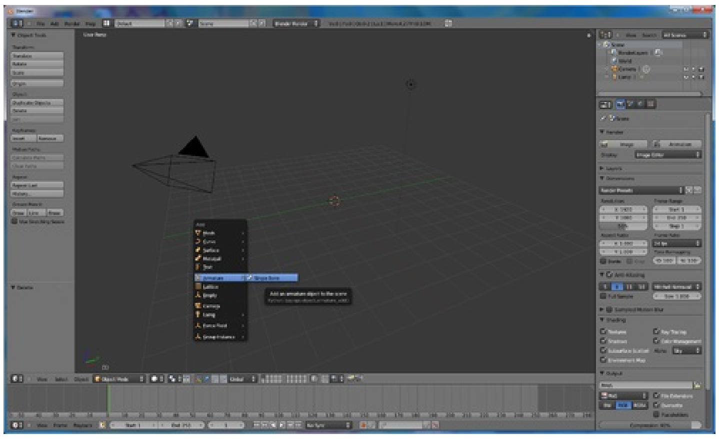
Your first armature

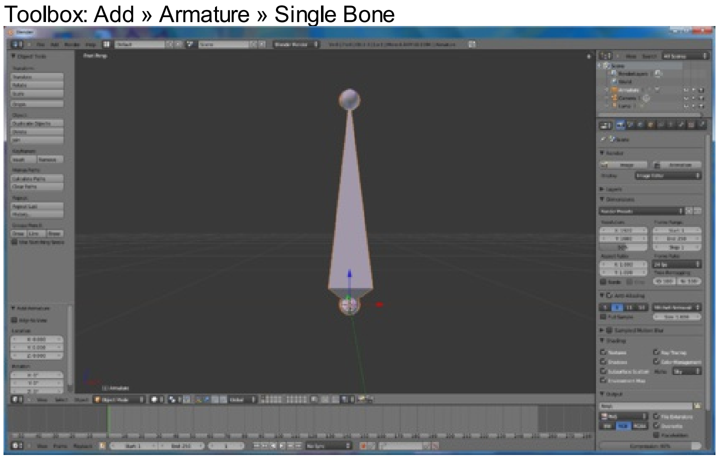
In order to see what we're talking about, let's try to add the default armature in Blender.

(Note that armature editing details are explained in the armatures editing section).

Open a default scene, then:

- delete all objects in the scene
- make sure the cursor is in the world origin with ⇧ ShiftC
- press 1 NumPad to see the world in Front view
- then, either:
    - in the Main Menu, Go to Add > Armature > Single Bone
    - -or- in the 3D view, add an armature with ⇧ ShiftA » Armature » Single Bone
- press Del numpad to see the armature at maximum zoom


Toolbox: Add » Armature » Single Bone


The default armature

The armature object

As you can see, an armature is like any other object type in Blender:

- It has a center, a position, a rotation and a scale factor.
- It has an ObData datablock, that can be edited in Edit mode.
- It can be linked to other scenes, and the same armature data can be reused on multiple objects.
- All animation you do in Object mode is only working on the whole object, not the armature’s bones (use the Pose mode to do this).

As armatures are designed to be posed, either for a static or animated scene, they have a specific state, called “rest position”. This is the armature’s default “shape”, the default position/rotation/scale of its bones, as set in Edit mode.

In Edit mode, you will always see your armature in rest position, whereas in Object and Pose mode, you usually get the current “pose” of the armature (unless you enable the Rest Position button of the Armature panel).

Armature chapter overview

In the "Armatures" section, we will only talk about armatures themselves, and specifically we will talk about:

- the armature object panels

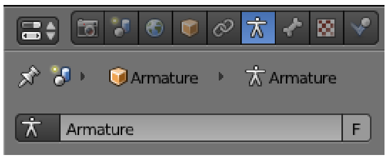
- the basics of bones
- the different armature visualizations
- the armature structure types
- how to select its parts,
- how to edit an armature
- how to Edit Bones
- how to edit bones properties
- how to sketch armatures with the Etch-a-Ton tool
- how to use templates

Armature Panels Overview

Mode: Object mode, Edit mode and Pose mode

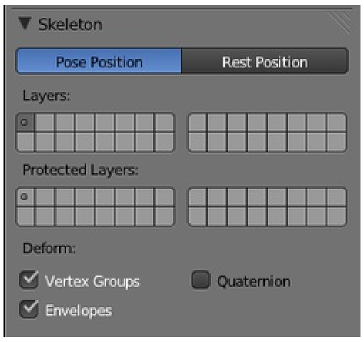
Panel: All in Properties window, Object data property

Let’s first have a general overview of the various panels gathering the armature settings, in Properties window, Object data context:



The Object data property in the Properties window.

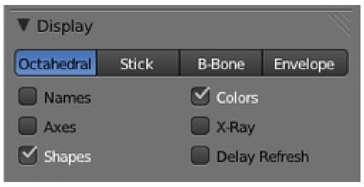
Skeleton panel (all modes)



The Skeleton panel.

In this panel you can arrange sets of bones into different layers for easier manipulation.

Display panel (all modes)

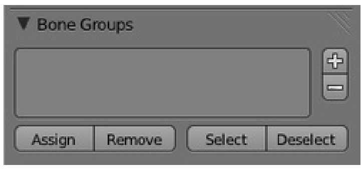


The Display panel.

This controls the way the bones appear in 3D view; you have 4 different options you can select.

There are several other options available which we will cover later on.

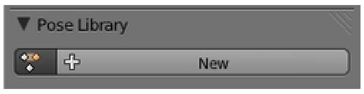
Bone groups panel (pose mode)



The Bone Groups panel.

Lets you assign sets of bones into groups for easy manipulation and management.

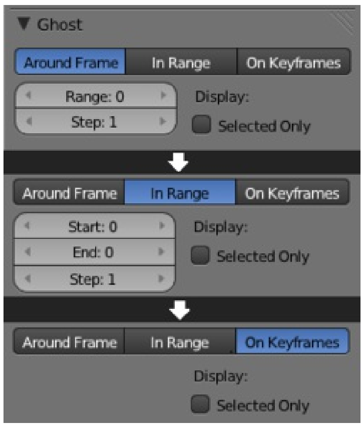
Pose Library panel (Pose mode)



The Pose Library panel.

Allows you to save different settings (location, rotation, scale) for selected bones for later use.

Ghost panel (all modes)



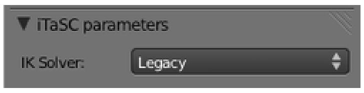
Range: 0
Step: 1

Start: 0
End: 0
Step: 1

The Ghost panel.

Allows you to see a set of different consecutive poses, very useful when animating.

iTaSC parameters panel (all modes)

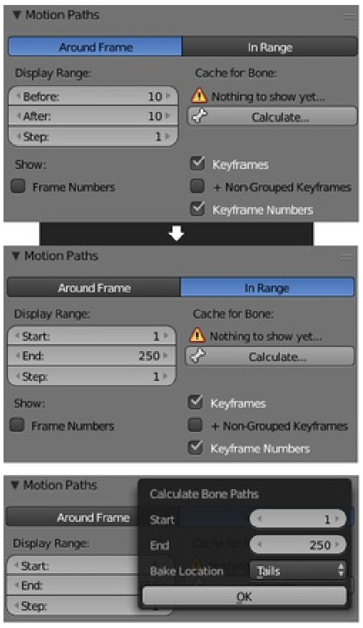


IK Solver: Legacy

The iTaSC parameters panel.

Defines the type of IK solver used in your animation.

Motion Paths panel (Pose mode)



Before: 10
After: 10
Step: 1
Frame Numbers
✓ Keyframes
+ Non-Grouped Keyframes
✓ Keyframe Numbers

Start: 1
End: 250
Step: 1

Start 1
End 250
Bake Location Tails

The Motion Paths panel.

In this panel you can enable visualization of the motion path your skeleton leaves when animated.

Custom Properties panel (all modes)

The Custom Properties panel.

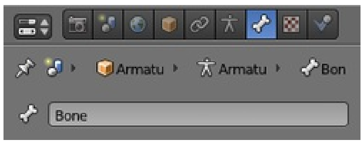
Panel for defining custom properties; this is used when scripting.

Bone Panels Overview

Mode: Object mode, Edit mode and Pose mode

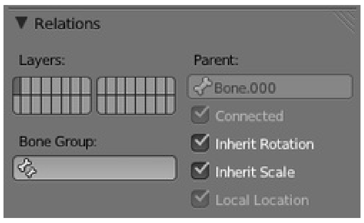
Panel: All in Properties window, Bone property

Let’s first have a general grasp of the various panels gathering the bone settings, in Properties window, Bone context:



The Bone context.

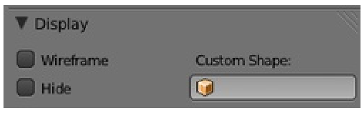
Relations panel (edit mode)



The Relations panel.

In this panel you can arrange sets of bones in different layers for easier manipulation.

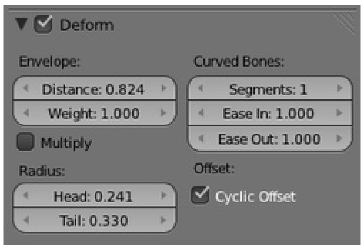
Display panel (object mode)



The Display panel.

Display panel lets you customize the look of your bones taking the shape of a another existing object.

Deform panel (all modes)



The Deform panel.

In this panel you can set basic properties of the bones.

Turning the Deform option on and off, includes the active bone in the Automatic Weight Calculation when the Mesh is Parented to the Armature using the Armature Deform with the "With Automatic Weights" option.

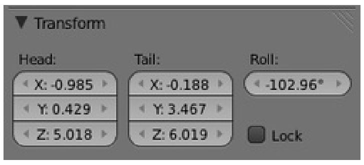
Also it's worth noting that by turning off a bone's deform option, makes it not influence the mesh at all, overriding any weights that it might have been assigned before; It mutes its influence.

Custom Properties panel (all modes)

The Custom Properties panel.

Panel for defining custom properties, this is used when scripting.

Transform panel (edit and pose mode)

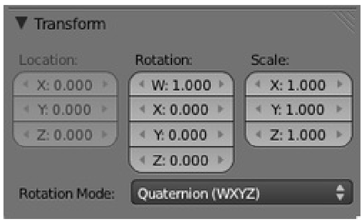

Transform

Head: X: -0.985 Y: 0.429 Z: 5.018
Tail: X: -0.188 Y: 3.467 Z: 6.019
Roll: -102.96°
Lock

The Transform panel(edit mode).

When in edit mode you can use this panel to control position and roll of individual bones.

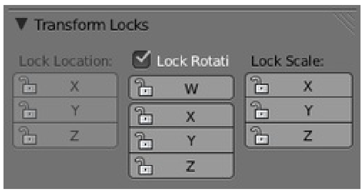
When in pose mode you can only set location for the main bone, and you can now set rotation and scale.


Transform

Location: X: 0.000 Y: 0.000 Z: 0.000
Rotation: W: 1.000 X: 0.000 Y: 0.000 Z: 0.000
Scale: X: 1.000 Y: 1.000 Z: 1.000
Rotation Mode: Quaternion (WXYZ)

The Transform panel(pose mode).

Transform Locks panel (pose mode)

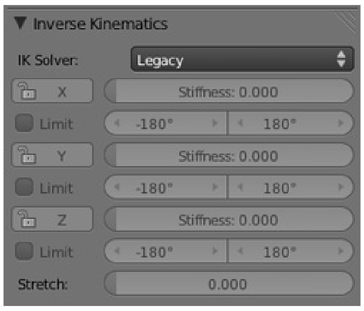

Transform Locks

Lock Location: X Y Z
Lock Rotati ✓: W X Y Z
Lock Scale: X Y Z

The Transform Locks panel.

This panel appears only in pose mode and allows you to restrict position, rotation and scale by axis on each bone in the armature.

Inverse Kinematics panel (pose mode)

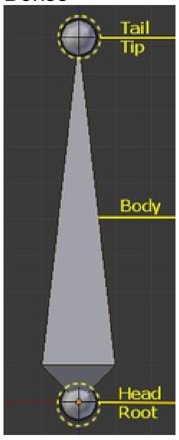

Inverse Kinematics

IK Solver: Legacy
X Stiffness: 0.000
Limit -180° 180°
Y Stiffness: 0.000
Limit -180° 180°
Z Stiffness: 0.000
Limit -180° 180°
Stretch: 0.000

The Inverse Kinematics panel.

This panel controls the way a bone or set of bones behave when linked in an inverse kinematic chain.

Bones



The elements of a bone.

Bones are the base elements of armatures.

They have three elements:

- the “start point” named **root** or **head**,
- the “body” itself,
- and the “end point” named **tip** or **tail**.

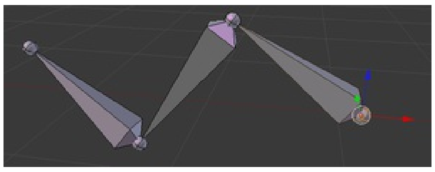
Select the default armature and press ⇆ Tab to enter Edit mode. As you can see, in this mode you can select the root and the tip, and move them as you do with mesh vertices (don't lose too much time here though, specific pages about selecting and editing will come later).

Both root and tip (the “ends”) define the bone by their respective position.

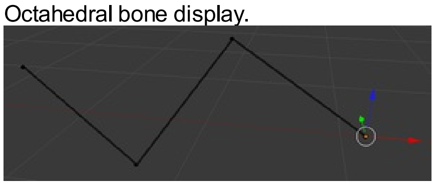
They also have a radius property, only useful for the envelope deformation method (see below).

Bones Visualization

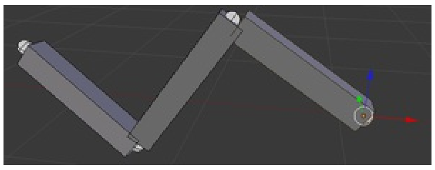
Bones can be visualized in various ways: Octahedron, Stick, B-Bone, Envelope and Wire. Custom shapes can be used, too!



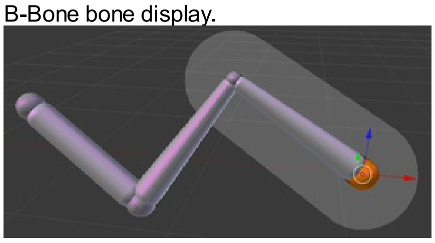
Octahedral bone display.



Stick bone display.



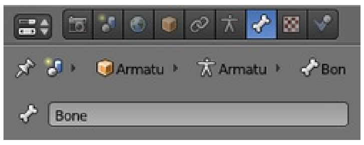
B-Bone bone display.



Envelope bone display.

Since armatures are made of bones, you’ll find more about this when we’ll talk about Armatures Visualization.

Activating Axes checkmark on the Armature/Display panel, will show local axes for each bone's tip. The Y axis is always aligned along the bone, oriented from root to tip. So, this is the "roll" axis of the bones.



The Bone context.

Bones properties

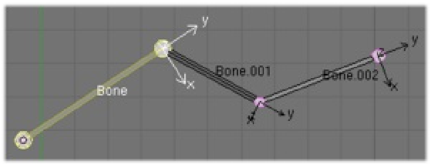
When bones are selected (hence in Edit mode and Pose mode), their properties are shown in the Bone button context of the Properties window.

This shows different panels used to control features of each selected bone; the panels change depending on which mode you’re working in.

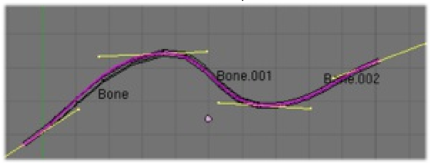
Bones Rigidity

Even though bones are rigid (i.e. behave as rigid sticks), they are made out of segments. Segments are small, rigid linked elements that can rotate between each other. By default, each new bone has only one segment and as such it cannot “bend” along its length. It is a rigid bone.

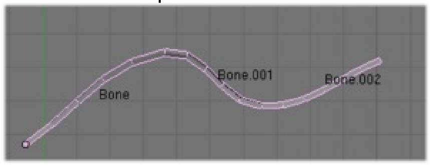
You can see these segments in Object mode and in Pose mode, and only if bones are visualized as B-bones; while in Edit mode bones are always drawn as rigid sticks. Note that in the special case of a single bone, you can't see these segments in Object mode, because they're aligned.



An armature of B-Bones, in Edit mode



The Bézier curve superposed to the chain, with its handles placed at bones’ ends.



The same armature in Object mode

When you connect bones to form a chain, Blender calculates a Bezier curve passing through all the bones' ends, and bones' segments in the chain will bend and roll to follow this invisible curve.

You have no direct access to this curve; you can only control it to some extent using bone properties, as explained in the editing pages.

In An armature of B-Bones in Edit mode we connected 3 bones, each one made of 5 segments. These are B-bones but as you see, in Edit mode they are shown as rigid elements. Look at The same armature in Object mode: now, in Object mode, we can see how the bones' segments smoothly “blend” into each other, even for roll.

Of course, a geometry influenced by the chain is smoothly deformed according to the Bezier curve! In fact, smooth bones are an easy way to replace long chains of many small rigid bones posed using IK...

However, if the chain has an influence on objects rather than geometry, the segments' orientation is not taken in account (details are explained in the skinning part).

When not visualized as B-Bones, bones are always shown as rigid sticks, *even though the bone segments are still present and effective* (see skinning to ObData).

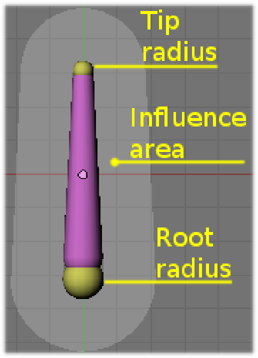
This means that even in e.g. Octahedron visualization, if some bones in a chain have several segments, they will nonetheless smoothly deform their geometry...

Bones influence

Basically, a bone controls a geometry when vertices "follow" the bone. This is like how the muscles and skin of your finger follow your finger-bone when you move a finger.

To do this, you have to define **how much** a bone influences a certain vertex.

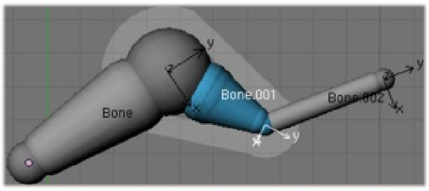
The simplest way is to have each bone affecting those parts of the geometry that are within a given range from it. This is called the *envelope technique*, because each bone can control only the geometry “enveloped” by its own influence area.



A bone in Envelope visualization, in Edit mode.

If a bone is visualized as Envelope, in Edit mode and in Pose mode you can see the area of influence, which depends on:

- the distance property
- the root's radius and the tip's radius.

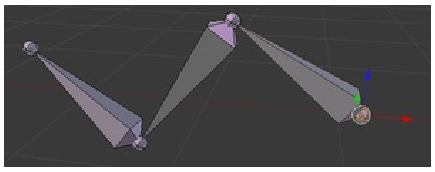


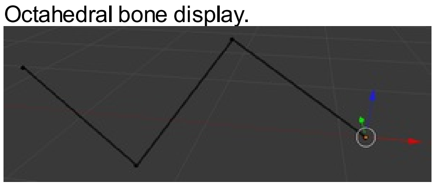
Our armature in Envelope visualization, in Pose mode.

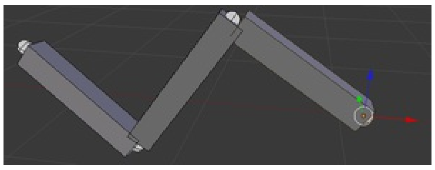
All these influence parameters are further detailed in the skinning pages.

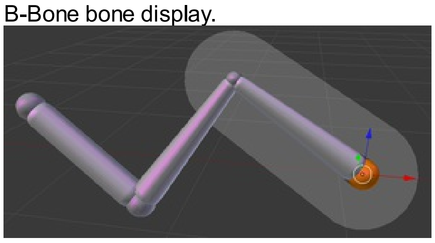
Armature visualization

We have 5 basic bone visualization: Octahedral, Stick, B-Bone, Envelope and Wire:


Octahedral bone display.


Stick bone display.


B-Bone bone display.


Envelope bone display.

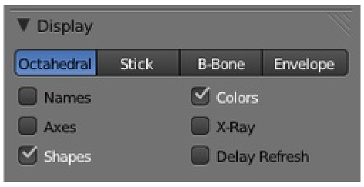
Wire bone display.

Display Panel

Mode: Object, Edit and Pose modes

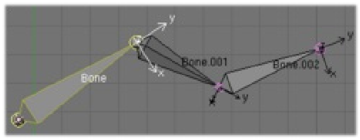
Panel: Display Object Data context

But let's first see some general visualization properties of armatures, found in the Display panel of the Object data context.



The Display panel.

Bone types

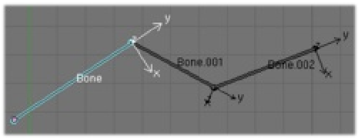


A basic armature in Octahedron visualization, Edit mode.
Note the **40°** rolled `Bone.001` bone.

Octahedral bone

This is the default visualization, well suited for most of editing tasks. It materializes:

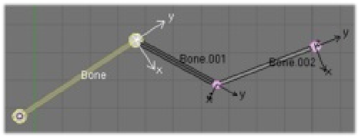
- The bone root (“big” end) and tip (“small” end).
- The bone “size” (its thickness is proportional to its length).
- The bone roll (as it has a square section).



The same armature in Stick visualization, Pose mode.
Note that `Bone.001` roll angle is not visible (except by its XZ axes).

Stick bone

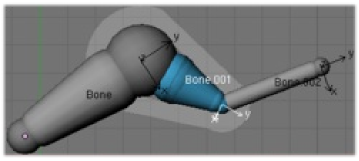
This is the simplest and most non-intrusive visualization. It just materializes bones by sticks of constant (and small) thickness, so it gives you no information about root and tip, nor bone size or roll angle.



The same armature in B-Bone visualization, Edit mode.

B-Bone bone

This visualization shows the curves of “smooth” multi-segmented bones; see the bone page for details.



The Envelope visualization.

Envelope bone

This visualization materializes the bone deformation influence. More on this in the bone page.

Attributes

Names
: When enabled, the name of each bone is drawn.

Colors
: This is only relevant for Pose mode, and is described in detail there.

Axes
: When enabled, the (local) axes of each bone are drawn (only relevant for Edit and Pose modes).

X-Ray
: When enabled, the bones of the armature will always be drawn on top of the solid objects (meshes, surfaces, …) – i.e. they will always be visible and selectable (this is the same option as the one found in the Display panel of the Object data context. Very useful when not in Wireframe mode.

Shapes
: When enabled, the default standard bone shape is replaced, in Object and Pose modes, by the shape of a chosen object (see below for details).

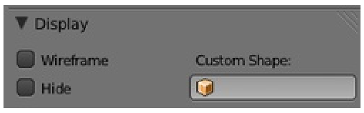
Delay Refresh
: When enabled, the bone doesn't deform its children when manipulating the bone in pose mode.

Shaped Bones

Mode: Object and Pose modes

Panel: Display panel from Bone context.

Blender allows you to give to each bone of an armature a specific shape (in Object and Pose modes), using another object as “template”. First of all, you have to enable the Shapes button (Armature panel).



The Display panel.

Attributes

Wireframe
: When enabled, bone is displayed in wireframe mode regardles of the viewport drawing mode. Useful for non-obstructive custom bone chains.

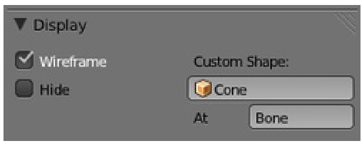
Hide
: Bone is not visible when not in Edit mode.

Custom Shape
: Object that defines the custom shape of the selected bone.

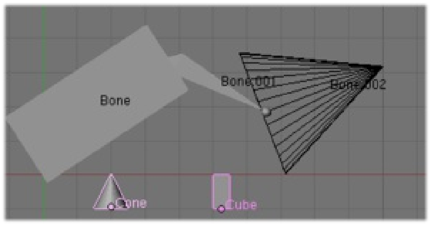
Custom At
: Bone that defines the display transform of this shape bone

To assign a custom shape to a bone, you have to:

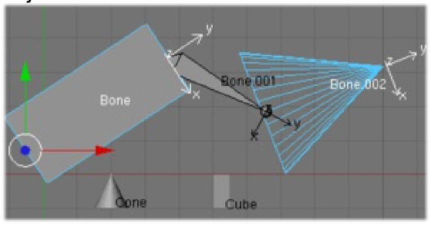
- Switch to Pose mode (Ctrl⇆ Tab).
- Select the relevant bone (RMB click on it).
- Go to the Display panel Custom Shape field and select the 3D object previously created in the scene; in this example we are using a cube and a cone. Tou can optionally set the At field to another bone.



The Display panel.



The armature with shapes assigned to two bones, in Object mode.
Note the centers of the `Cone` and `Cube` objects.



The same armature in Pose mode…

Note that:

- These shapes will never be rendered – like any bone, they are only visible in 3D views.
- Even if any type of object seems to be accepted by the OB field (meshes, curves, even metas…), only meshes really work – all other types just make the bone invisible; nothing is drawn…
- The center of the shape object will be at the *root of the bone* (see the bone page for root/tip).
- The object properties of the shape are ignored (i.e. if you make a parallelepiped out of a cube by modifying its dimensions in Object mode, you’ll still have a cube shaped bone…).
- The “along bone” axis is the Y one, and the shape object is always scaled so that one Blender Unit stretches along the whole bone length.
- If you need to remove the custom shape of the bone, just right click in the Custom Shape field and select Reset to default value in the popup menu.

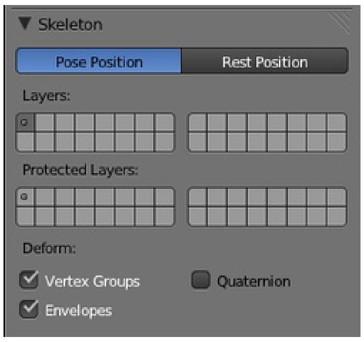
So to summarize all this, you should use meshes as shape objects, with their center at their lower-Y end, and an overall Y length of **1.0**

BU.

Armature Layers

Mode: Object, Edit and Pose modes

Panel: Skeleton panel, Object data context



The Skeleton panel.

Each armature has 32 “Armature layers” which allow you to organize your armature by “regrouping” sets of bones into layers; this works similar to scene layers (those containing your objects). You can then “move” a bone to a given layer, hide or show one or several layers, etc.

Showing/hiding bone layers

Only bones in active layers will be visible/editable – but they will always be effective (i.e move objects or deform geometry), whether in an active layer or not. To (de)activate a layer, you have several options, depending in which mode you are in:

- In all modes, use the row of small buttons at the top of the Display Options group, Armature panel. If you want to enable/disable several layers at once, as usual, hold ⇧ Shift while clicking…
- In Edit and Pose modes, you can also do this from the 3D Views, either by using the menu (Armature » Switch Armature Layers or Pose » Switch Armature Layers), or the ⇧ ShiftM shortcut, to display a small pop-up dialog containing the same buttons as described above (here again, you can use ⇧ Shift LMB clicks to (de)select several layers at once).

Protected Layers

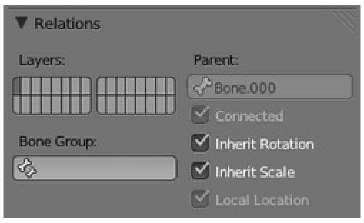
You can lock a given bone layer for all proxies of your armature, i.e. all bones in this layer won’t be editable. To do so, in the Skeleton panel, Ctrl LMB click on the relevant button, the layer lock will be enabled.

Protected layers in proxy are restored to proxy settings on file reload and undo.

Bone Layers

Mode: Object, Edit and Pose modes

Panel: Relations panel Bone context



The Relations panel.

Moving bones between layers

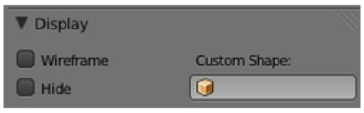
Obviously, you have to be in Edit or Pose modes to move bones between layers – note that as with objects, bones can lay in several layers at once, just use the usual ⇧ Shift LMB clicks… First of all, you have to select the chosen bone(s)!

- In the Button window, use the “layer buttons” of each selected bone “sub-panel” (Armature Bones panel) to control in which layer(s) it lays.
- In the 3D View window, use the menu (Armature » Move Bone To Layer or Pose » Move Bone To Layer) or hit M to show the usual pop-up layers dialog. Note that this way, *you assign the same layers to all selected bones.*

Hiding Bones

Mode: Edit and Pose modes

Panel: Display panel, Bone context



The Display panel.

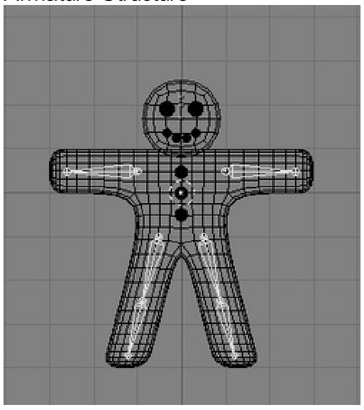
You do not have to use bone layers to show/hide some bones. As with objects, vertices or control points, you can use the H key:

- H will hide the selected bone(s).
- ⇧ ShiftH will hide all bones *but the selected one(s)*.
- AltH will show all hidden bones.

You can also use the Hide check button of the Display panel, Bone context).

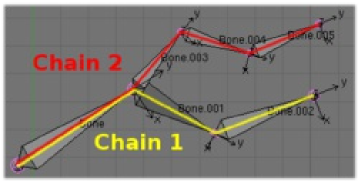
Note that hidden bones are specific to a mode – i.e. you can hide some bones in Edit mode, they will still be visible in Pose mode, and vice-versa. Hidden bone in Pose mode are also invisible in Object mode. And in Edit mode, the bone to hide must be fully selected, not just his root or tip…

Armature Structure


The very basic armature of the Gingerbread Man tutorial.

Armatures mimic real skeletons. They are made out of bones, which are (by default) rigid elements. But you have more possibilities than with real skeletons: In addition to the “natural” rotation of bones, you can also translate and even scale them! And your bones do not have to be connected to each other; they can be completely free if you want. However, the most natural and useful setups imply that some bones are related to others, forming so-called “chains of bones”, which create some sort of “limbs” in your armature, as detailed below.

Chains of Bones



An armature with two chains of bones.

The bones inside an armature can be completely independent from each other (i.e. the modification of one bone does not affect the others). But this is not often a useful set up: To create a leg, all bones “after” the thigh bone should move “with” it in a well-coordinated manner. This is exactly what happens in armatures – by parenting a bone to the next one in the limb, you create a “chains of bones”. These chains can be ramified. For example, five fingers attached to a single “hand” bone.

Bones are chained by linking the tip of the parent to the root of the child. Root and tip can be *connected*, i.e. they are always exactly at the same point; or they can be *free*, like in a standard parent-child object relationship.

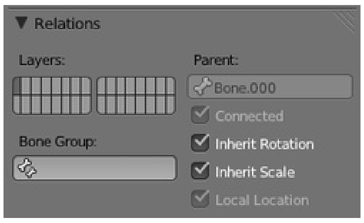
A given bone can be the parent of several children, and hence be part of several chains at the same time.

The bone at the beginning of a chain is called its *root bone*, and the last bone of a chain is the *tip bone* (don’t confuse them with similar names of bones’ ends!).

Chains of bones are a particularly important topic in posing (especially with the standard *forward kinematics* versus “automatic” *inverse kinematics* posing techniques). You create/edit them in Edit mode, but except in case of connected bones, their relationships have no effect on bone transformations in this mode (i.e. transforming a parent bone won’t affect its children).

Editing Bones Relationships

This is detailed in the editing pages, but let us have a quick look at this important feature.



The Armature Bones panel with two bones selected, and their Child of settings highlighted.

The easiest way to manage bones relationships is to use the Relations panel Bone context:

- First, select the bones you want to edit (selection order does not matter here).
- To *parent* a bone to another one, select the name of this parent in its drop-down Parent list.
- To *unparent* a bone, just select the void entry in the same Parent list.
- To *connect* a bone to its parent, enable its small Con button.
- To *unconnect* a bone, disable its Con button.

Selecting armature's bones

Mode: Edit mode

Panel: Bone panel

You can select and edit bones of armatures in Edit mode and in Pose mode. Here, we will see how to select bones in Edit mode. Selecting bones in Pose mode is similar to selecting in Edit mode with a few specific differences that will be detailed in the posing part.

Similar to vertices/edges selection in meshes, there are two ways to select whole bones in Edit mode:

- directly, by selecting the bone's body
- selecting both of its end points (root and tip)

This is an important point to understand, because selecting bones' ends only might lead to non-obvious behavior, with respect to which bone you actually select, see the .

Note that unlike the mesh draw type the armature draw type has no effect on selection behavior. In other words, you can select a bone's end or body the same way regardless of the bone visualization chosen.

Selecting bones' ends

To select bones' ends you have the standard selection methods.

action	shortcut	menu	mouse
Select a bone's end			RMB -click on it
Add or Remove from the current selection			⇧ Shift RMB
(De)select the ends of all bones	A	Select » Select/Deselect All	
Invert the current selection	CtrlI	Select » Inverse	
Box selection tool ON	B	Select » Border Select	
Box selection			click and drag LMB the box around the ends you want to add to the current selection click and drag LMB to remove from the current selection release LMB to validate hit Esc or click RMB to cancel
Box selection tool OFF	B or Esc		RMB
Lasso selection			click and drag Ctrl LMB the lasso around the ends you want to add to the current selection click and drag Ctrl⇧ Shift LMB to remove from the current selection release LMB to validate hit Esc or click RMB to cancel

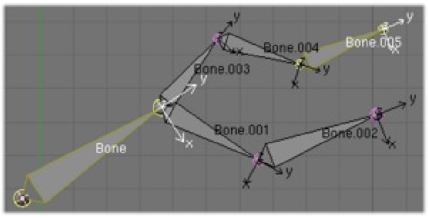
Inverse selection

As stated above, you have to remember that these selection tools are for bones' ends only, not the bones' bodies.

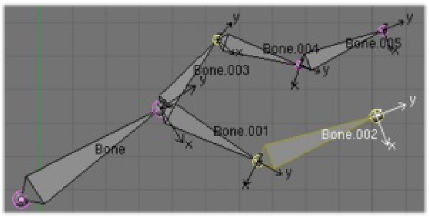
For example, the Inverse selection option (CtrlI) inverts the selection of bones' ends, not of bones (see *Inverse selection*).

Remember that a bone is selected only if both its ends are selected. So, when the selection status of bones' ends is inverted, a new set of bones is selected.

Inverse selection



Two bones selected.



The result of the inverse selection (CtrlI): the bones' ends selection has been inverted, and not the bones selection…

Selecting connected bones' ends

Another example is: when you select the root of a bone connected to its parent, you also implicitly select the tip of its parent (and vice versa).

Remember: when selecting bones' ends, the tip of the parent bone is the "same thing" as the root of its children bones.

Selecting Bones

By RMB -clicking on a bone's body, you will select it (and hence you will implicitly select its root and tip).

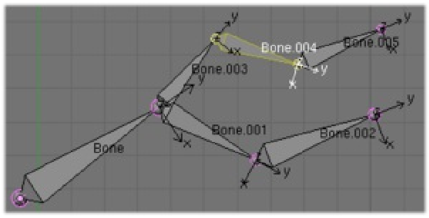
To each selected bone corresponds a sub-panel in the Armature Bones panel (Editing context, F9). These sub-panels contain settings for some of the bones' properties (regarding e.g. relationships between bones, bones' influence on deformed geometry, etc.), as we will see later.

Using ⇧ Shift RMB , you can add to/remove from the selection.

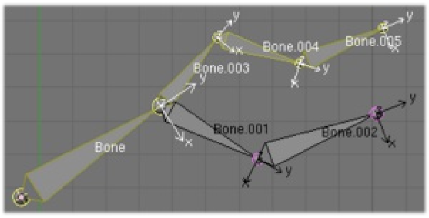
You also have some **advanced selection** options, based on their relations.

You can select at once all the bones in the chain which the active (last selected) bone belongs to by using the *linked selection* tool, L.

Linked bones selection



A single selected bone.



Its whole chain selected with L.

You can deselect the active bone and select its immediate parent or one of its children using respectively Select » Select Parent ([) or Select » Select Child (]). If you prefer to keep the active bone in the selection, use Select » Extend Select Parent (Ctrl[) or Select » Extend Select Child (Ctrl]).

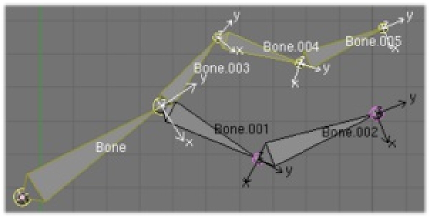
Deselecting connected bones

There is a subtlety regarding connected bones.

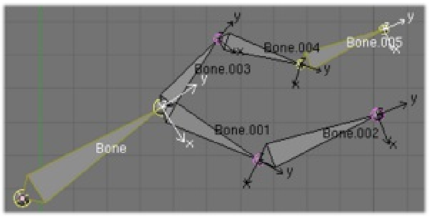
When you have several connected bones selected, if you deselect one bone, *you will in fact deselect its tip, but not its root if it is also the tip of another selected bone.*

To understand this, look at *Bone deselection in a selected chain*.

Bone deselection in a selected chain



A selected chain.



After ⇧ Shift RMB -clicking `Bone.003`

After ⇧ Shift RMB -clicking `Bone.003`:

- `Bone.003`'s tip (which is same as `Bone.004`'s root) is deselected
- `Bone` is `Bone.003`'s parent. Therefore `Bone.003`'s root is same as the tip of `Bone`. Since `Bone` is still selected, its tip is selected. Thus the root of `Bone.003` remains selected.

Armature Editing

Mode: Edit mode

Hotkey: ⇆ Tab

As with any other object, you edit your armature in Edit mode (⇆ Tab).

Editing an armature means two main domains of action:

- Editing the bones – i.e. adding/inserting/deleting/extruding/sub-dividing/joining them…
- Editing the bones' properties – this includes key features, like transform properties (i.e. grab, scale, etc…) and relationships between bones (parenting and connecting), as well as bones' names, influence, behavior in Pose mode, etc.

These are standard editing methods, quite similar for example to meshes editing. Blender also features a more advanced "armature sketching" tool, called Etch-a-Ton. The same tool might also be used in templating, i.e. using another armature as template for the current one…

One important thing to understand about armature editing is that you **edit the rest position of your armature**, i.e. its "default state". An armature in its *rest position* has all bones with no rotation and scaled to **1.0** in their own local space.

The different poses you might create afterwards are based on this rest position – so if you modify it in Edit mode, all the poses already existing will also be modified. Thus you should in general be sure that your armature is definitive before starting to skin and pose it!

Please note that some tools work on bones' ends, while others work on bones themselves. Be careful not to get confused.

Editing Bones

Mode: Edit mode

Hotkey: ⇆ Tab

You'll learn here how to add, delete or subdivide bones. We will also see how to prevent any bone transformation in Edit mode, and the option that features an automatic mirroring of editing actions along the X axis.

Adding Bones

To add bones to your armature, you have more or less the same options as when editing meshes:

- Add menu,
- extrusion,
- Ctrl LMB clicks,
- fill between joints,
- duplication.

Add Menu

Mode: Edit mode

Hotkey: ⇧ ShiftA

In the 3D view, ⇧ ShiftA » Bone to add a new bone to your armature.

This bone will be:

- of one Blender Unit of length,
- oriented towards the positive Y axis of the view,
- with its root placed at the 3D cursor position,
- with no relationship with any other bone of the armature.

Extrusion

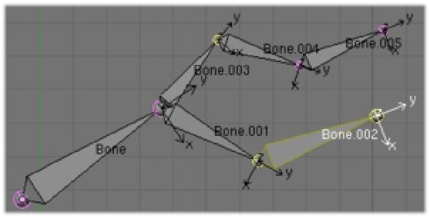
Mode: Edit mode

Hotkey: E, ⇧ ShiftE

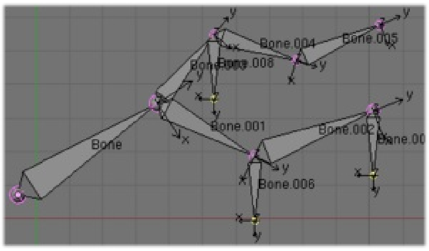
Menu: Armature » Extrude

When you press the E key, for each selected tip (either explicitly or implicitly), a new bone is created. This bone will be the child of "its" tip owner, and connected to it. As usual, once extrusion is done, only the new bones' tips are selected, and in grab mode, so you can place them to your liking. See (*Extrusion example*).

Extrusion example



An armature with three selected tips.

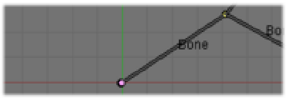


The three extruded bones.

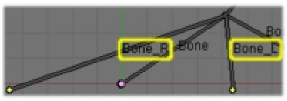
You also can use the rotating/scaling extrusions, as explained for meshes here, by hitting respectively ER and ES – as well as "locked" extrusion along a global or local axis.

Bones have an extra "mirror extruding" tool, called by hitting ⇧ ShiftE. By default, it behaves exactly like the standard extrusion. But once you have enabled the X-Axis mirror editing option (see below), each extruded tip will produce *two new bones*, having the same name except for a leading "_L/_R" code (for left/right, see the next page). The "_L" bone behaves like the single one produced by the default extrusion – you can grab/rotate/scale it exactly the same way. The "_R" bone is its mirror counterpart (along the armature's local X axis), see (*Mirror extrusion example*).

Mirror extrusion example



A single selected bone's tip.



The two mirror-extruded bones.

!

Note that exactly as in mesh editing, if you press Esc right after you have pressed E, the extruded bones will be there but their length will be zero, so very likely this will give you some headaches. If you realize the problem immediately, you can undo by pressing CtrlZ.

In case you're wondering, you cannot just press X to solve this as you would in mesh editing, because extrusion selects the newly created tips, and as explained below the delete command ignores bones' ends. To get rid of these extruded bones without undoing, you would have to move the tips, then select the bones and delete them.

Mouse Clicks

Mode: Edit mode

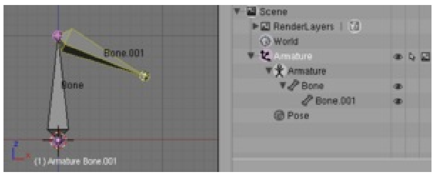
Hotkey: Ctrl LMB

If at least one bone is selected, Ctrl LMB -clicking adds a new bone.

About the new bone's tip:

- after you Ctrl LMB -clicked it becomes the active element in the armature,
- it appears to be right where you clicked, but…
- …(as in mesh editing) it will be on the plane parallel to the view and passing through the 3D cursor.

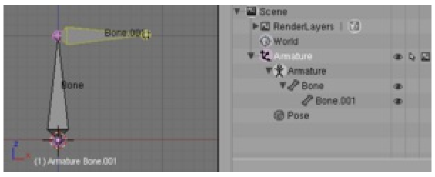
The position of the root and the parenting of the new bone depends on the active element:



Ctrl-clicking when the active element is a **bone**

If the active element is a **bone**

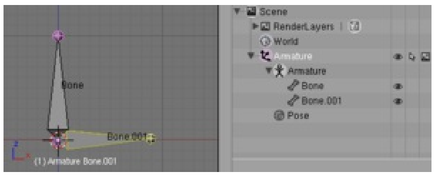
- the new bone's root is placed on the active bone's tip
- the new bone is parented and connected to the active bone (check the outliner in *Ctrl-clicking when the active element is a bone*).



Ctrl-clicking when the active element is a **tip**

If the active element is a **tip**:

- the new bone's root is placed on the active tip
- the new bone is parented and connected to the bone owning the active tip (check the outliner in *Ctrl-clicking when the active element is a tip*).

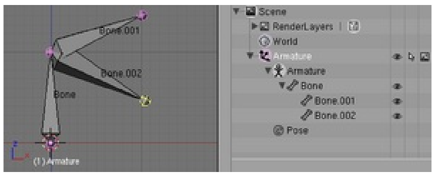


Ctrl-clicking when the active element is a **disconnected root**

If the active element is a **disconnected root**:

- the new bone's root is placed on the active root
- the new bone is **NOT** parented to the bone owning the active root (check the outliner in *Ctrl-clicking when the active element is a disconnected root*).

And hence the new bone will **not** be connected to any bone.



Ctrl-clicking when the active element is a **connected root**

If the active element is a **connected root**:

- the new bone's root is placed on the active root

- the new bone **IS** parented and connected to the parent of the bone owning the active root (check the outliner in *Ctrl-clicking when the active element is a connected root*).

This should be obvious because if the active element is a connected root then the active element is also the tip of the parent bone, so it is the same as the second case.

As the tip of the new bone becomes the active element, you can repeat these ctrl-clicks several times, to consecutively add several bones to the end of the same chain.

Fill between joints

Mode: Edit mode

Hotkey: F

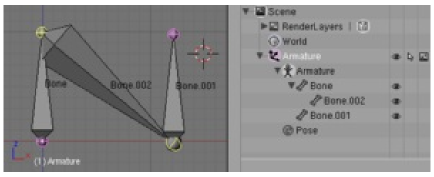
Menu: Armature » Fill Between Joints

The main use of this tool is to create one bone between two selected ends by pressing F, similar to how in mesh editing you can “create edges/faces”.

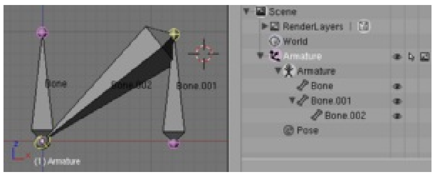
If you have one root and one tip selected, the new bone:

- will have the root placed on the selected tip
- will have the tip placed on the selected root
- will be parented and connected to the bone owning the selected tip

Fill between a tip and a root



Active tip on the left

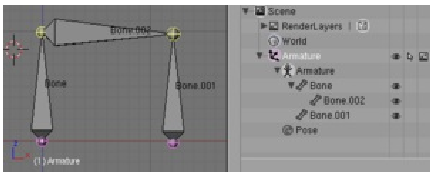


Active tip on the right

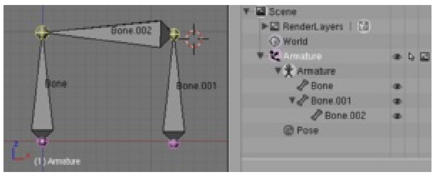
If you have two tips selected, the new bone:

- will have the root placed on the selected tip closest to the 3D cursor
- will have the tip placed on the other selected tip
- will be parented and connected to the bone owning the tip used as the new bone's root

Fill between tips



3D cursor on the left



3D cursor on the right

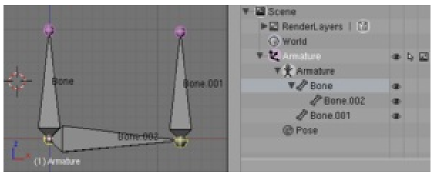
If you have two roots selected, you will face a small problem due to the event system in Blender not updating the interface in real time.

When clicking F, similar to the previous case, you will see a new bone:

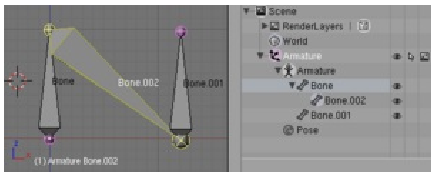
- with the root placed on the selected root closest to the 3D cursor
- with the tip placed on the other selected root
- parented and connected to the bone owning the root used as the new bone's root

If you try to move the new bone, Blender will update the interface and you will see that the new bone's root moves to the tip of the parent bone.

Fill between roots



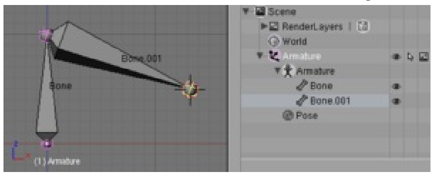
Before UI update (3D cursor on the left)

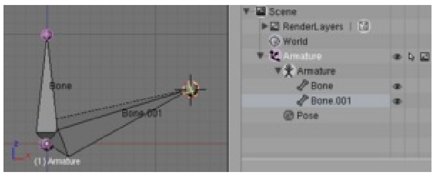


After UI update, correct visualization

Clicking F with only one bone end selected will create a bone from the selected end to the 3D cursor position, and it won't parent it to any bone in the armature.

Fill with only one bone end selected


Fill with only one tip selected


Fill with only one root selected

You will get an error when:

- trying to fill two ends of the same bone, or
- trying to fill more than two bone ends.

Duplication

Mode: Edit mode

Hotkey: ⇧ ShiftD

Menu: Armature » Duplicate

This tool works on selected bones; selected ends are ignored.

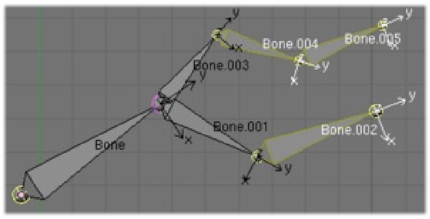
As in mesh editing, by pressing ⇧ ShiftD:

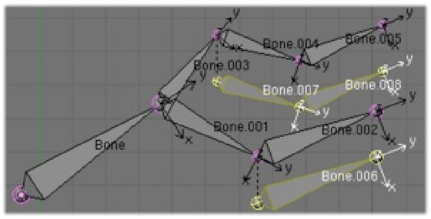
- the selected bones will be duplicated,
- the duplicates become the selected elements and they are placed in grab mode, so you can move them wherever you like.

If you select part of a chain, by duplicating it you'll get a copy of the selected chain, so the copied bones are interconnected exactly like the original ones.

The duplicate of a bone which is parented to another bone will also be parented to the same bone, even if the root bone is not selected for the duplication. Be aware, though, that if a bone is parented **and connected** to an unselected bone, its copy will be parented **but not connected** to the unselected bone (see *Duplication example*).

Duplication example


An armature with three selected bones and a selected single root.


The three duplicated bones. Note that the selected chain is preserved in the copy, and that `Bone.006` is parented but not connected to `Bone.001`, as indicated by the black dashed line. Similarly, `Bone.007` is parented but not connected to `Bone.003`.

Deleting Bones

You have two ways to remove bones from an armature: the standard deletion, and merging several bones in one.

Standard deletion

Mode: Edit mode

Hotkey: X

Menu: Armature » Delete

This tool works on selected bones: selected ends are ignored.

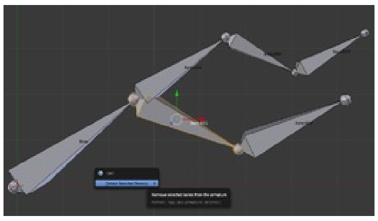
To delete a bone, you can:

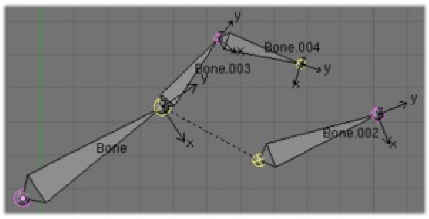
- press the standard X key and confirm, or

- use the menu Armature » Delete and confirm.

If you delete a bone in a chain, its child(ren) will be automatically re-parented to its own parent, **but not connected**, to avoid deforming the whole armature.

Deletion example


An armature with two selected bones, just before deletion.



The two bones have been deleted. Note that `Bone.002`, previously connected to the deleted `Bone.001`, is now parented but not connected to `Bone`.

Merge

Mode: Edit mode

Hotkey: AltM

Menu: Armature » Merge

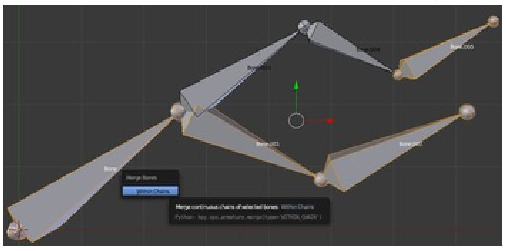
You can merge together several selected bones, *as long as they form a chain*. Each sub-chain formed by the selected bones will give one bone, whose root will be the root of the root bone, and whose tip will be the tip of the tip bone.

Confirm by clicking on Within Chains in the Merge Selected Bones pop-up.

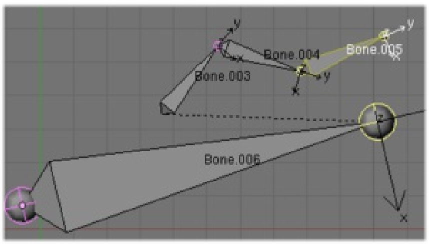
If another (non-selected) chain origins from inside of the merged chain of bones, it will be parented to the resultant merged bone. If they were connected, it will be connected to the new bone.

Here's a strange subtlety (see *Merge example*): even though connected (the root bone of the unmerged chain has no root sphere), the bones are not visually connected – this will be done as soon as you edit one bone, differently depending in which chain is the edited bone (compare the bottom two images of the example to understand this better).

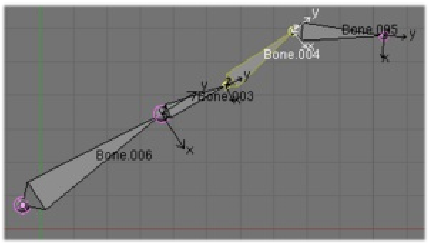
Merge example



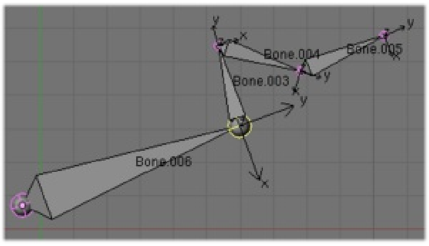
An armature with a selected chain, and a single selected bone, just before merging.



Bones `Bone`, `Bone.001` and `Bone.002` have been merged in `Bone.006`, whereas `Bone.005` wasn't modified. Note `Bone.003`, connected to `Bone.006` but not yet "really" connected.



`Bone.004` has been rotated, and hence the tip of `Bone.006` was moved to the root of `Bone.003`.



The tip of `Bone.006` has been translated, and hence the root of `Bone.003` was moved to the tip of `Bone.006`...

Subdividing Bones

Mode: Edit mode

Hotkey: W1, W2

Menu: Armature » Subdivide, Armature » Subdivide Multi

You can subdivide bones, to get two or more bones where there was just one bone. The tool will subdivide all selected bones, preserving the existing relationships: the bones created from a subdivision always form a connected chain of bones.

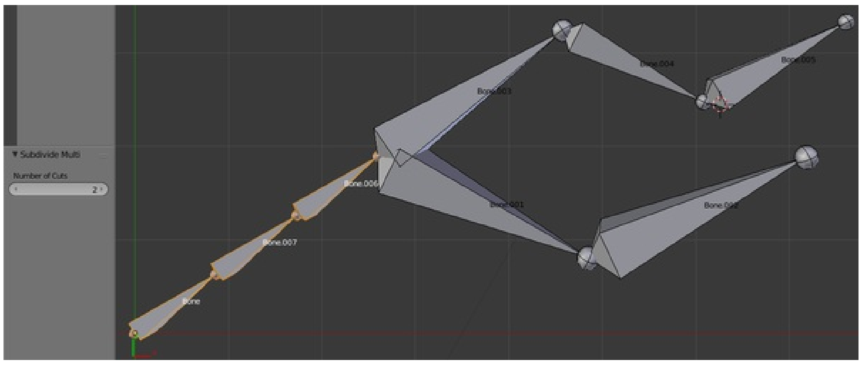
To create two bones out of each selected bone:

- press W » Subdivide, same as W1, or
- select Armature » Subdivide from the header menu

To create an arbitrary number of bones from each selected bone:

- in Subdivide options on the Toolshelf in Subdivide Multi section specify the number of cuts you want. As in mesh editing, if you set `n` cuts, you'll get `n+1` bones for each selected bone.

Subdivision example



The selected bone has been "cut" two times, giving three sub-bones.

Locking Bones

You can prevent a bone from being transformed in Edit mode in several ways:

- The active bone can be locked clicking on the Lock button in Transform panel of the Bone context;
- All selected bones can be locked pressing ⇧ ShiftW » Toggle Bone Options » Lock or
- select Armature » Bone Settings » Lock) in 3D-header.

If the root of a locked bone is connected to the tip of an unlocked bone, it won't be locked, i.e. you will be able to move it to your liking. This means that in a chain of connected bones, when you lock one bone, you only really lock its tip. With unconnected bones, the locking is effective on both ends of the bone.

X-Axis Mirror Editing

Another very useful tool is the X-Axis Mirror editing option (Tool panel > Armature Options, while Armature is selected in Edit Mode), working a bit like the same mesh editing tool. When you have pairs of bones of the same name with just a different "side suffix" (e.g. `.R`/`.L`, or `_right`/`_left`…), once this option is enabled, each time you transform (move/rotate/scale…) a bone, its "other side" counterpart will be transformed accordingly, through a *symmetry along the armature local X axis*. As most rigs have at least one axis of symmetry (animals, humans, …), it's an easy way to spare you half of the editing work! See also next page for more on naming bones.

Separating Bones in a new Armature

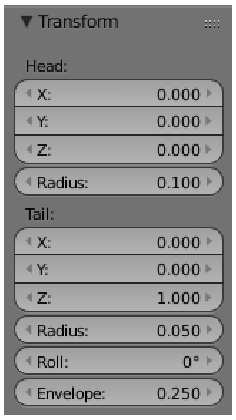
You can, as with meshes, separate the selected bones in a new armature object (Armature » Separate, CtrlAltP) – and of course, in Object mode, you can join all selected armatures in one (Object » Join Objects, CtrlJ).

Editing Bone Properties

In this page, you will learn how to edit and control most of the properties for Blender bones – For editing bones in an armature, you should read the previous page first! We will see how to manage the bones' relationships, rename them, etc.

Transforming Bones

We won't detail here the various transformations of bones, nor things like axis locking, pivot points, and so on, as they are common to most object editing, and already described here (note however that some options, like snapping, do not seem to work, even though they are available…). The same goes for mirroring, as it's nearly the same as with mesh editing. Just keep in mind that bones' roots and tips behave more or less like meshes' vertices, and bones themselves act like edges in a mesh.

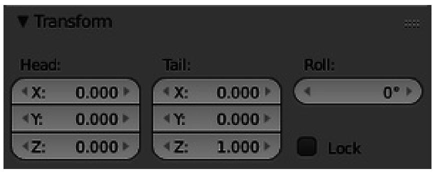

Transform

Head:
X: 0.000
Y: 0.000
Z: 0.000
Radius: 0.100

Tail:
X: 0.000
Y: 0.000
Z: 1.000
Radius: 0.050
Roll: 0°
Envelope: 0.250

The Transform Properties panel for armatures in Edit mode.

As you know, bones can have two types of relationships: They can be parented, and in addition connected. Parented bones behave in Edit mode exactly as if they had no relations – you can grab, rotate, scale, etc. a parent bone without affecting its descendants. However, connected bones must always have parent's tips connected to child's roots, so by transforming a bone, you will affect all its connected parent/children/siblings.


Transform

Head: X: 0.000 Y: 0.000 Z: 0.000
Tail: X: 0.000 Y: 0.000 Z: 1.000
Roll: 0°
Lock

The Transform panel of Bone context for armatures in Edit mode.

Finally, you can edit in the Transform Properties panel (N) the positions and radius of both ends of the active selected bone, as well as its roll rotation.

Scaling Specificities in B-Bone and Envelope visualization

Mode: Edit mode and Pose mode, Envelope and B-Bone visualization

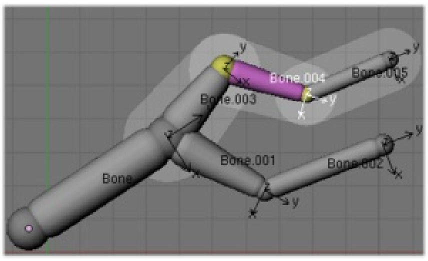
Hotkey: CtrlAltS

Menu: Armature » Transform » Scale Envelope Distance, Armature » Transform » Scale BBone

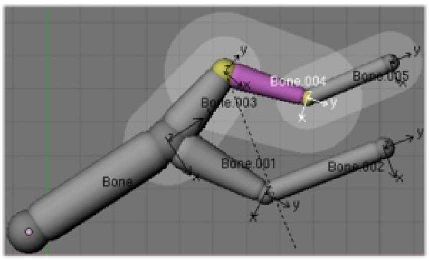
CtrlAltS activates a transform tool that is specific to armatures. It has different behavior depending on the active visualization, as explained below:

In Envelope visualization, it allows you to edit the influence of the selected bones (their Dist property, see the skinning part) – as with the "standard" scaling with this visualization (see the previous section), this is a one-value property, so there is no axis locking and such.

Envelope scaling example



A single bone selected in Envelope visualization.

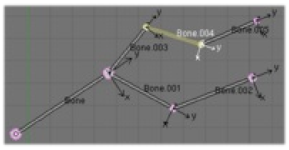


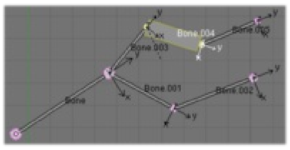
Its envelope scaled with CtrlAltS.

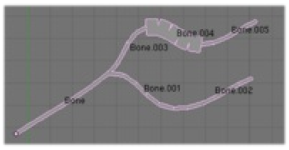
In the other visualizations, it allows you to edit the “bone size”. This seems to only have a visible effect in B-Bone visualization, but is available also with Octahedron and Stick… This tool in this situation has another specific behavior: While with other transform tools, the “local axes” means the object’s axes, here they are the bone’s own axes (when you lock to a local axis, by pressing the relevant key twice, the constraint is applied along the selected bone’s local axis, not the armature object’s axis).

WARNING! If you have more than one bone selected, using this tool crashes Blender!

“Bone size” scaling example


A single “default size” bone selected in B-Bone visualization.


Its size scaled with CtrlAltS.


The same armature in Object mode and B-Bone visualization, with `Bone.004`’s size scaled up.

Scaling Specificities in Envelope Visualization

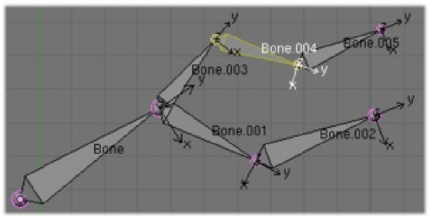
Mode: Edit mode, Envelope visualization

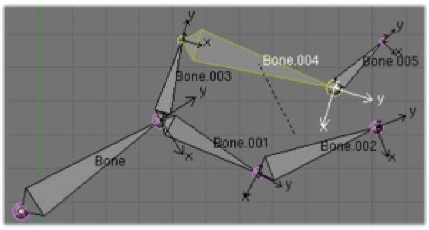
Hotkey: AltS

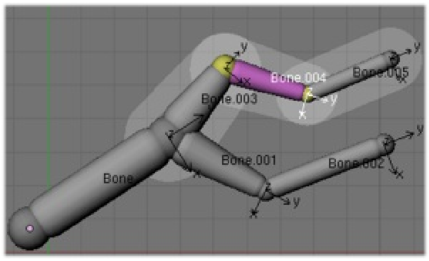
Menu: Armature » Transform » Scale Radius

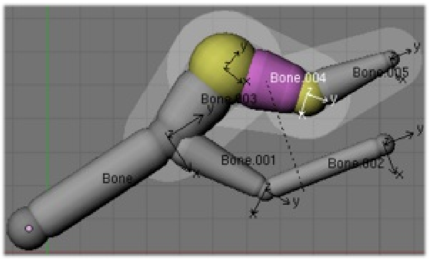
When bones are displayed using Octahedron, Stick or B-Bone visualizations, scaling will behave as expected, similar to scaling mesh objects. When bones are displayed using Envelope visualization, scaling will have a different effect: it will scale the radius of the selected bones’s ends. (see: skinning part). As you control only one value (the radius), there is no axis locking here. And as usual, with connected bones, you scale at the same time the radius of the parent’s tip and of the children’s roots.

Scaling of a bone in Octahedron and Envelope visualizations.


A single selected bone…


…Scaled in Octahedron visualization.


A single selected bone…


…Scaled in Envelope visualization – its length remains the same, but its ends’ radius are bigger.

Note that when you resize a bone (either by directly scaling it, or by moving one of its ends), Blender automatically adjusts the end-radii of its envelope proportionally to the size of the modification. Therefore, it is advisable to place all the bones first, and only then edit these properties.

Bone Direction

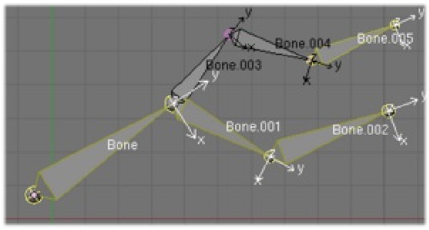
Mode: Edit mode

Hotkey: AltF, W2

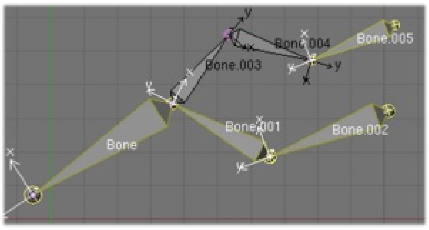
Menu: Armature » Switch Direction, Specials » Switch Direction

This tool is available from the Armature menu (AltF) and from the Specials pop-up menu(W). It allows you to switch the direction of the selected bones (i.e. their root will become their tip, and vice versa).

Switching the direction of a bone will generally break the chain(s) it belongs to. However, if you switch a whole (part of a) chain, the switched bones will still be parented/connected, but in “reversed order”. See the *Switching example*.



An armature with one selected bone, and one selected chain of three bones, just before switching.



The selected bones have been switched. `Bone.005` is no more connected nor parented to anything. The chain of switched bones still exists, but reversed (Now `Bone.002` is its root, and `Bone` is its tip). `Bone.003` is now a free bone.

Switching example.

Bone Roll

Mode: Edit mode

Hotkey: CtrlR, CtrlN

Menu: Armature » Bone Roll » …

In Edit mode, you have options dedicated to the control of the bone roll rotation (i.e. the rotation around the Y axis of the bone). Each time you add a new bone, its default roll is so that its Z axis is as perpendicular to the current 3D view as possible. And each time you transform a bone, Blender tries to determine its best roll…

But this might lead to an unclear armature, with bones rolled in all angles… nasty! To address this problem, you have three options:

- Armature » Bone Roll » Set Roll (CtrlR) will start a roll-specific rotation, which behaves like any other transform operations (i.e. move the mouse and LMB click to validate, or type a numeric value and hit enter – or RMB click or hit Esc to cancel everything).
- Armature » Bone Roll » Recalculate Roll (CtrlN):

Type

Local X Tangent
: Sets the selected bone roll so that its Z axis is as much as possible pointed to local X tangent.

Local Z Tangent
: Sets the selected bone roll so that its Z axis is as much as possible pointed to local Z tangent.

Global X Axis/Global Y Axis/Global Z Axis
: Sets the selected bone roll so that its Z axis is as much as possible pointed to global X, Y or Z axis respectively.

Active Bone
: Sets the selected bone roll so that its Z axis is as much as possible pointed to active (selected) other bone.

View Axis
: Sets the selected bone roll so that its Z axis is as much as possible pointed toward to 3D-View plane.

Cursor
: Sets the selected bone roll so that their Z axis is as much as possible pointed to the 3D cursor.

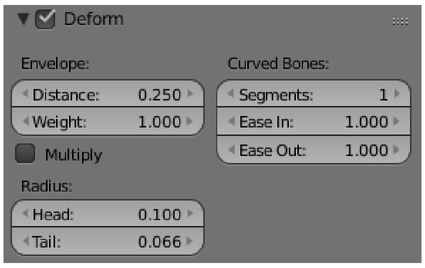
Flip Axis
: Negate flipping the alignment axis.

Shortest Rotation
: Ignore the axis direction, use the shortest rotation to align.

Deform properties for bone

Mode: Edit and Pose mode

Panel: Deform panel of Bones context


Deform

Envelope:
Distance: 0.250
Weight: 1.000
Multiply

Radius:
Head: 0.100
Tail: 0.066

Curved Bones:
Segments: 1
Ease In: 1.000
Ease Out: 1.000

The Deform panel of Bones context in Edit mode.

Most bones' properties (excepted the transform ones) are regrouped in each bone's sub-panel, in the Deform panel (Bones context. Let's detail them.

Note that some of them are also available in the 3D views, through the three pop-up menus Toggle Setting (⇧ ShiftW or Armature » Bone Settings » Toggle a Setting), Enable Setting (Ctrl⇧ ShiftW or Armature » Bone Settings » Enable a Setting), and Disable Setting (AltW or Armature » Bone Settings » Disable a Setting) – all three have the same entries, their respective effect should be obvious…

These settings specify how armature bones and particularly their ends influence on model geometry in the posing workflow.

Deform settings with bone envelope visualization

Envelope

Distance
: Sets overall bone deformation distance to geometry.

Weight
: Allow to set bone weight value to geometry.

Multiply
: When deforming bones, multiply effects of Vertex Group weights with Envelope influence.

Radius

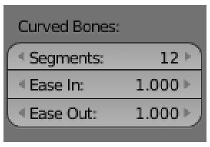
Head
: Sets radius of head (root) of bone (only with envelope visualization) and respectively envelope influence in its area.

Tail
: Radius of tail (tip) of bone (only with envelope visualization) and respectively envelope influence in its area.

Bone rigidity settings

Mode: Pose modes (only for B-Bone visualization)

Panel: Deform panel of Bones context


Curved Bones:
Segments: 12
Ease In: 1.000
Ease Out: 1.000

Rigitity settings in in Pose mode.

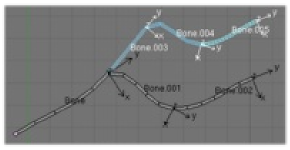
Even though you have the Segments setting available in Edit mode (in the Deform panel of Bone context), you should switch to the Pose mode (Ctrl⇆ Tab) to edit these "smooth" bones' properties – one explanation to this strange need is that in Edit mode, even in B-Bone visualization, bones are drawn as sticks, so you can't visualize the effects of these settings.

Curved Bones

Segments
: This setting controls the number of subdivisions that a bone has.

Easy In
: Length of first virtual Bezier handle with B-Bone visualization.

Easy Out
: Length of second virtual Bezier handle with B-Bone visualization.



An armature in Pose mode, B-Bone visualization: `Bone.003` has one segment, `Bone.004` has four, and `Bone.005` has sixteen.

We saw in this page that bones are made of small rigid segments mapped to a "virtual" Bézier curve. The Segm numeric field allows you to set the number of segments inside a given bone – by default, it is **1**, which gives a standard rigid bone! The higher this setting (max **32**), the smoother the bone, but the heavier the pose calculations…

Each bone's ends are mapped to its "virtual" Bézier curve's "auto" handle. Therefore, you can't control their direction, but you can change their "length" using the In and Out numeric fields, to control the "root handle" and "tip handle" of the bone, respectively. These values are proportional to the default length, which of course automatically varies depending on bone length, angle with previous/next bones in the chain, and so on.

Chain Relations Editing

Mode: Edit mode

Panel: Relations panel of Bone context

Hotkey: CtrlP, AltP

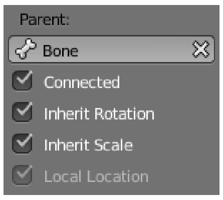
Menu: Armature » Parent » …

You can edit the relationships between bones (and hence create/modify the chains of bones) both from the 3D views and the Properties window. Whatever method you prefer, it's always a matter of deciding, for each bone, if it has to be parented to another one, and if so, if it should be connected to it.

To parent and/or connect bones, you can:

- In a 3D view, select the bone and *then* its future parent, and hit CtrlP (or Armature » Parent » Make Parent…). In the small Make Parent menu that pops up, choose Connected if you want the child to be connected to its parent, else click on Keep Offset. If you have selected more than two bones, they will all be parented to the last selected one. If you only select one already-parented bone, or all selected bones are already parented to the last selected one, your only choice is to connect them, if not already done. If you select only one non-parented bone, you'll get the Need selected bone(s) error message…

  With this method, the newly-children bones won't be scaled nor rotated – they will just be translated if you chose to connect them to their parent's tip.



The bones chain relations editing settings.

Parent

Parent bone field
: Parent bone in same armature.

Conected
: When bone has a parent, bone's head is stuck to the parent's tail.

Inherit Rotation
: Bone inherits rotation from parent bone.

Inherit Scale
: Bone inherits scaling from parent bone.

Local Location
: Bone location is set in local space.

Object Children (only in Pose mode)

Relative Parenting
: Object children will use relative transform, like deform.

- In the Bone context, Relations panel, for each selected bone, you can select its parent in the Parent drop-down list to the upper right corner of its sub-panel. If you want them to be connected, just enable the little Conected button to the below of the list.

  With this method, the tip of the child bone will never be translated – so if Conected is enabled, the child bone will be completely transformed by the operation.

To disconnect and/or free bones, you can:

- In a 3D view, select the desired bones, and hit AltP (or Armature » Parent » Clear Parent…). In the small Clear Parent menu that pops up, choose Clear Parent to completely free all selected bones, or Disconnect Bone if you just want to break their connections.
- In the Buttons window, Armature Bones panel, for each selected bone, you can select no parent in the Parent drop-down list of its sub-panel, to free it completely. If you just want to disconnect it from its parent, disable the Con button.

Note that relationships with non-selected children are never modified.

Naming Bones

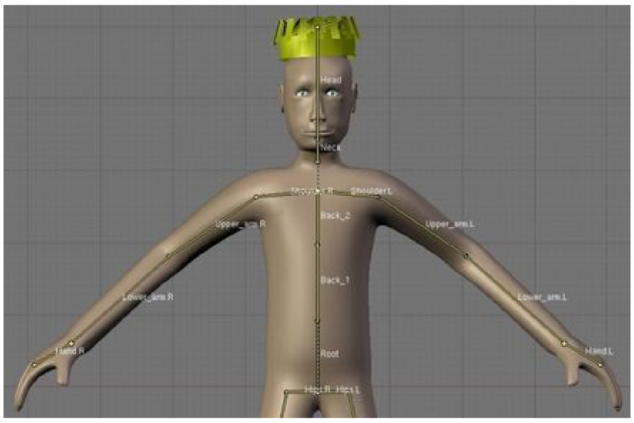
Mode: Edit and Pose mode

Panel: Bone context, Item section of Properties panel in 3D views (N))

You can rename your bones, either using the Bone field in Item section of the Properties panel in the 3D views (N), or using the top field of the Bone context for the active bone.

Blender also provides you some tools that take advantage of bones named in a left/right symmetry fashion, and others that automatically name the bones of an armature. Let's look at this in detail.

Naming Conventions



An example of left/right bone naming in a simple rig.

Naming conventions in Blender are not only useful for you in finding the right bone, but also to tell Blender when any two of them are counterparts.

In case your armature can be mirrored in half (i.e. it’s bilaterally symmetrical), it’s worthwhile to stick to a left/right naming convention. This will enable you to use some tools that will probably save you time and effort (like the X-Axis Mirror editing tool we saw above…).

- First you should give your bones meaningful base-names, like `leg`, `arm`, `finger`, `back`, `foot`, etc.
- If you have a bone that has a copy on the other side (a pair), like an arm, give it one of the following separators:
  - Left/right separators can be either the second position (`L_calfbone`) or last-but-one (`calfbone.R`)
  - If there is a lower or upper case `L`, `R`, `left` or `right`, Blender handles the counterpart correctly. See below for a list of valid separators. Pick one and stick to it as close as possible when rigging; it will pay off. For example:

Valid Separators.

Separator	example		
(nothing)	hand**Left**	→	hand**Right**
_ *(underscore)*	Hand**_L**	→	Hand**_R**
. *(point)*	hand**.l**	→	hand**.r**
- *(dash)*	Foot**-l**	→	Foot**-r**
(space)	pelvis **LEFT**	→	pelvis **RIGHT**

Note that all examples above are also valid with the left/right part placed before the name. You can only use the short `L`/`R` code if you use a separator (i.e. `handL`/`handR` won’t work!).

- Before Blender handles an armature for mirroring or flipping, it first removes the number extension, if it’s there (like `.001`)
- You can copy a bone named `bla.L` and flip it over using W » Flip Names. Blender will name the copy `bla.L.001` and flipping the name will give you `bla.R`.

Bone name flipping

Mode: Edit mode

Hotkey: W4

Menu: Armature » Flip Names

You can flip left/right markers (see above) in selected bone names, using either Armature » Flip Names, or Specials » Flip Names (W4). This can be useful if you have constructed half of a symmetrical rig (marked for a left or right side) and duplicated and mirrored it, and want to update the names for the new side. Blender will swap text in bone names according to the above naming conventions, and remove number extensions if possible.

Auto bone naming

Mode: Edit mode

Hotkey: W5, W6, W7

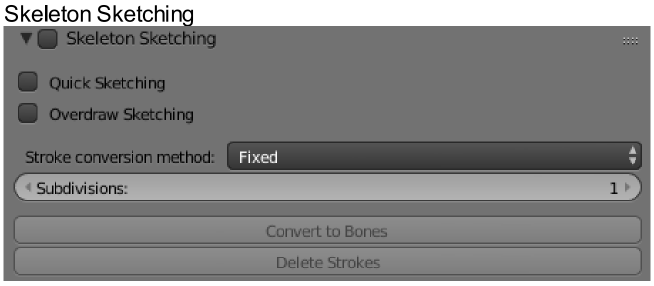
Menu: Armature » AutoName Left-Right, Armature » AutoName Front-Back, Armature » AutoName Top-Bottom

The three AutoName entries of the Armature and Specials (W) menus allows you to automatically add a suffix to all selected bones, *based on the position of their root relative to the armature center and its local coordinates*:

- AutoName Left-Right will add the `.L` suffix to all bones *with a positive X-coordinate root*, and the `.R` suffix to all bones *with a negative X-coordinate root*. If the root is exactly at **0.0** on the X-axis, the X-coordinate of the tip is used. If both ends are at **0.0** on the X-axis, the bone will just get a period suffix, with no L/R (as Blender cannot decide whether it is a left or right bone…).
- AutoName Front-Back will add the `.Bk` suffix to all bones *with a positive Y-coordinate root*, and the `.Fr` suffix to all bones *with a negative Y-coordinate root*. The same as with AutoName Left-Right goes for **0.0** Y-coordinate bones…

- AutoName Top-Bottom will add the `.Top` suffix to all bones *with a positive Z-coordinate root*, and the `.Bot` suffix to all bones *with a negative Z-coordinate root*. The same as with AutoName Left-Right goes for **0.0** Z-coordinate bones…

Skeleton Sketching



The Skeleton Sketching panel in its default (inactive) state.

If you think that creating a whole rig by hand, bone after bone, is quite boring, be happy: some Blender developers had the same feeling, and created the Skeleton Sketching tool, formerly the Etch-a-ton tool, which basically allows you to “draw” (sketch) whole chains of bones at once.

Skeleton Sketching is obviously only available in Edit mode, in the 3D views. You control it through its Skeleton Sketching panel in the Transform panel, which you can open with N. Use mouse (LMB to draw strokes, and RMB for gestures. Showing its tool panel won’t enable sketching – *you must tick the checkbox next to Skeleton Sketching to start drawing bone chains* (otherwise, you remain in the standard Edit mode…).

Sketching is done in two steps:

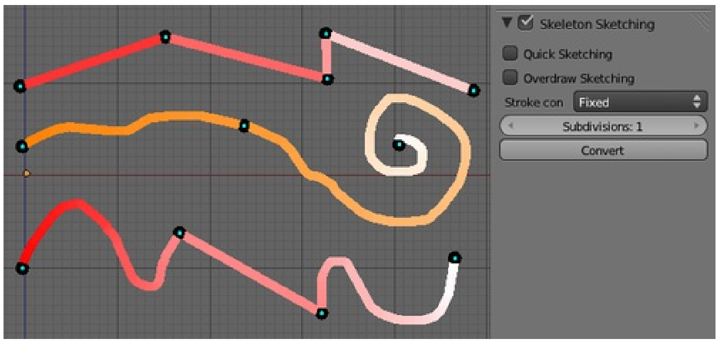
1. Drawing some “smooth” and/or polygonal lines (called “strokes”). Each stroke corresponds to a chain of bones.
2. Converting these strokes into real chains of bones, using different methods.

The point of view is important, as it determines the future bones’ roll angle*: the Z axis of a future bone will be aligned with the view Z axis of the 3D viewin which you drawits “parent” stroke (unless you use the Template converting method…). Strokes are drawn in the current viewplane passing through the 3D cursor, but you can create somewhat “3D” strokes using the Adjustdrawing option in different views (see below).*

If you enable the small Quick Sketch option, the two steps are merged into one: once you have finalized the drawing of a stroke (see below), it is immediately converted to bones (using the current active method) and deleted. This option makes bone sketching quick and efficient, but you lose all the advanced stroke editing possibilities…

Sketches are not saved into Blender files, so you can’t interrupt a sketching session without losing all your work! Note also that the *sketching is common to the whole Blender session*, i.e. there is only one set of strokes (one sketch) in Blender, and not one per armature, or even per file…

Drawing Chains

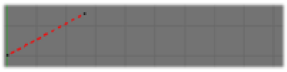


Strokes example. From top to bottom:
A selected polygonal stroke of four straight segments, oriented from left to right.
An unselected free stroke of two segments, oriented from left to right.
A mixed stroke, with one straight segment between two free ones, right to left.

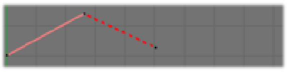
So, each stroke you draw will be a chain of bones, oriented from the starting point (the reddest or most orange part of the stroke) to its end (its whitest part). A stroke is made of several *segments*, delimited by small black dots – *there will be at least one bone per segment* (except with the Template conversion method, see next page), so all black points represents future bones’ ends. There are two types of segments, which can be mixed together:

Straight Segments

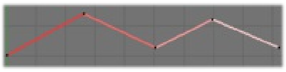
To create a straight segment, click LMB at its starting point. Then move the mouse cursor, without pressing any button – a dashed red line represents the future segment. Click LMB again to finalize it. *Each straight segment of a stroke will always create one and only one bone*, whatever convert algorithm you use (except for the Retarget conversion method).



The first segment has been started (LMB click) and the mouse moved to its end



The first segment has been finalized by a second LMB click, which also started a



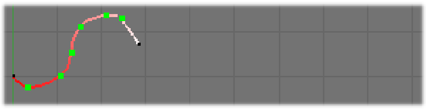
Repeating these steps, we now have a four-segment polygonal stroke.

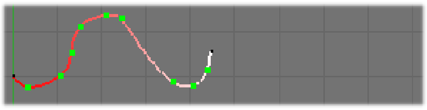
point.　　new segment…

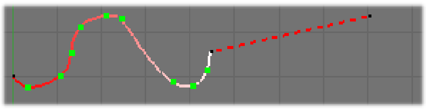
Free Segments

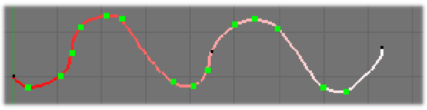
To create a free (curved) segment, click *and hold* LMB at its starting point. Then draw your segment by moving the mouse cursor – as in any paint program! Release LMB to finalize the segment – you will then be creating a new *straight* segment, so if you would rather start a new *free* segment, you must immediately re-press LMB… The free segments of a stroke will create different number of bones, in different manners, depending on the conversion method used. The future bones' ends for the current selected method are represented by small green dots for each one of those segments, *for the selected strokes only*.

> The free segment drawing uses the same Manhattan Dist setting as the grease pencil tool (User Preferences window, Edit Methods "panel", Grease Pencil group) to control where and when to add a new point to the segment – so if you feel your free segments are too detailed, raise this value a bit, and if you find them too jagged, lower it.


While drawing a first free segment (LMB click and drag).


The first free segment finalized (releasing LMB).


If you now move the mouse without pressing LMB again, you'll create a straight segment…


But if you immediately click again and drag LMB, you'll instead start a new free segment.

Drawing free segments example.

You finalize a whole stroke by clicking RMB. You can cancel the stroke you are drawing by hitting Esc. You can also snap strokes to underlying meshes by holding Ctrl while drawing. By the way, the Peel Objects button at the bottom of the Bone Sketching panel is the "same thing" as the "monkey" button of the snapping header bar controls shown when Volume snap element is selected – see the snap to mesh page for details.

Selecting Strokes

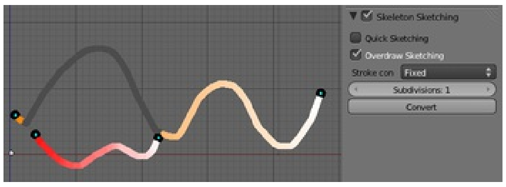
A stroke can be selected (materialized by a solid red-to-white line), or not (shown as a orange-to-white line) – see (*Strokes example*) above. As usual, you select a stroke by clicking RMB on it, you add one to/remove one from the current selection with a ⇧ Shift RMB click, and A (de)selects all strokes…

Deleting

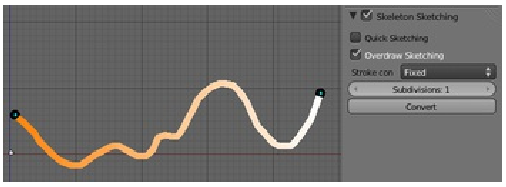
Hitting X or clicking on the Delete button (Bone Sketching panel) deletes the selected strokes (be careful, no warning/confirmation pop-up menu here…). See also the gesture description below.

Modifying Strokes

You can adjust, or "redraw" your strokes by enabling the Overdraw Sketching option of the Bone Sketching panel. This will modify the behavior of the strokes drawing (i.e. LMB clicks and/or hold): when you draw, you won't create a new stroke, but rather modify the nearest one. The part of the old stroke that will be replaced by the new one are drawn in gray. *This option does not take into account stroke selection*, i.e. all strokes can be modified this way, not just the selected ones… Note also that even if it is enabled, when you draw too far away from any other existing stroke, you won't modify any of them, but rather create a new one, as if Overdraw Sketching was disabled.



Adjusting a stroke: the gray part of the "unselected" (orange) stroke will be replaced by the currently drawn "replacement".



Stroke adjusted.

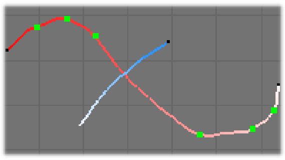
Adjusting stroke example.

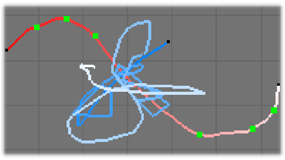
Finally, note that there is no undo/redo for sketch drawing…

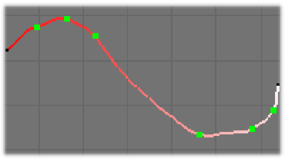
Gestures

There quite a few things about strokes editing that are only available through gestures. Gestures are started by clicking and holding ⇧ Shift+ LMB (when you are not already drawing a stroke…), and materialized by blue-to-white lines. A gesture can affect several strokes at once.

There is no direct way to cancel a gesture once you've started "drawing" it. So the best thing to do, if you change your mind (or made a "false move"), is to continue to draw until you get a disgusting scribble, crossing your stroke several times – in short, something that the gesture system would never recognize!

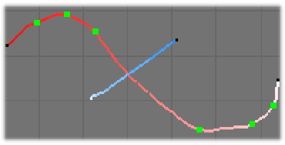

Damn! I didn’t want to cut this stroke here!

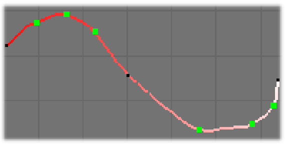

Let’s doodle a bit…


Phew! That was close, but the stroke is still in one piece…

Cut

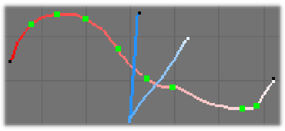
To **cut** a segment (i.e. add a new black dot inside it, making two segments out of one), “draw” a straight line crossing the chosen segment where you want to split it.

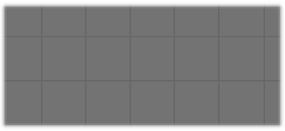

Gesture.


Result.

Delete

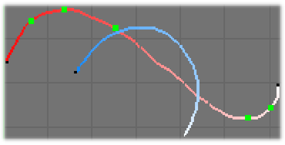
To **delete** a stroke, draw a “V” crossing the stroke to delete twice.

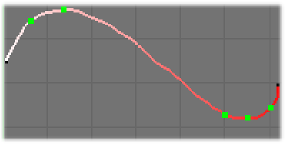

Gesture.


Result.

Reverse

To **reverse** a stroke (i.e. the future chain of bones will be reversed), draw a “C” crossing twice the stroke to reverse.


Gesture.


Result.

Converting to Bones

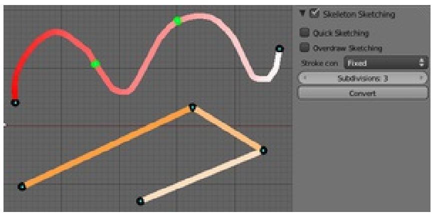
Once you have one or more selected strokes, you can convert them to bones, using either the Convert to Bones button of the Skeleton Sketching panel. Each selected stroke will generate a chain of bones, oriented from its reddest end to its whitest one. Note that converting a stroke does not delete it.

There are four different conversion methods – three “simple” ones, and one more advanced and complex, Retarget, that reuses bones from the same armature or from another one as a template for the strokes to convert, and which is detailed in the next page. Anyway, remember that *straight segments are always converted to one and only one bone* (except for the Template conversion method), and that the future bones’ ends are shown as green dots on selected free segments.

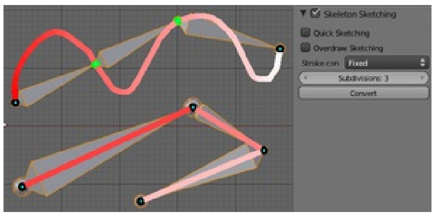
Remember also that the roll rotation of the created bones has been set during their “parent” stroke drawing (except for the Template conversion method) – their Z axis will be aligned with the view Z axis of the active 3D view at draw time.

Fixed

With this method, each free segment of the selected strokes will be uniformly divided in `n` parts (set in Num numeric field), i.e. will give `n` bones.



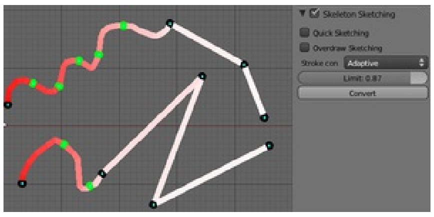
The Fixed conversion settings and its preview on selected strokes.



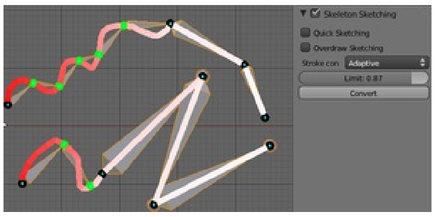
The Fixed conversion result.

Adaptative

With this method, each free segment of the selected strokes will create as many bones as necessary to follow its shape closely enough – this “closely enough” parameter being set by the Threshold numeric field; higher values giving more bones, following more closely the segments’ shape. So the more twisted a free segment, the more bones it will generate.



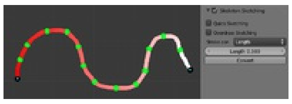
The Adaptative conversion settings and its preview on selected strokes.

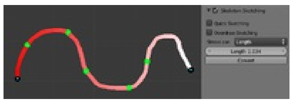


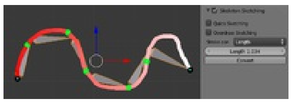
The Adaptative conversion result.

Length

With this method, each free segment of the selected strokes will create as many bones as necessary, so that none of them is longer than the Length numeric field value (in Blender Units).


The Length conversion settings and its preview on selected strokes.


Using a larger length value.


The Length conversion result.

Retarget

Retarget template bone chain to stroke.

Template
: Template armature that will be retargeted to the stroke. This is a more complex topic, detailed in its own page.

Retarget roll mode
: Method used to adjust the roll of bones when retargeting

    None
    : Don’t adjust roll.

    View
    : Roll bones to face the view.

    Joint
    : Roll bone to original joint plane offset.

Autoname
: Automatically generate values to replace &N and &S suffix placeholders in template names

Number
: Text to replace &N with (e.g. ‘Finger.&N’ -> ‘Finger.1’ or ‘Finger.One’)

Side
: Text to replace &S with (e.g. ‘Arm.&S’ -> ‘Arm.R’ or ‘Arm.Right’)

See more details about this method here.

Armature Templating

The idea of templating is to use an already existing armature as base ("template") to create a new armature. It differs from a simple copy in that you can directly define the new armature different in some aspects than its reference rig.

In Blender, the only templating tool is the bone sketching one (Skeleton Sketching (heretofore called Etch-a-ton), described in here), with its Retarget conversion method – so you should have read its page before this one!

Using Bone Sketching

Mode: Edit mode

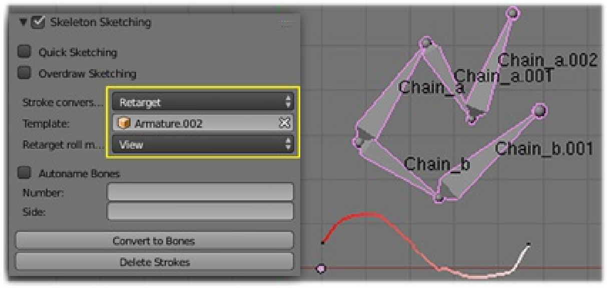
Panel: section Skeleton Sketching on the panel Properties (N)

The Retarget conversion method of Skeleton Sketching tool maps a copy of existing bones to each selected stroke. The new bones will inherit some of their properties (influence, number of segments, etc.) from the corresponding bones in the template, but they will acquire their lengths, rolls and rotation from the sketch; so these properties would be different as compared to the template.

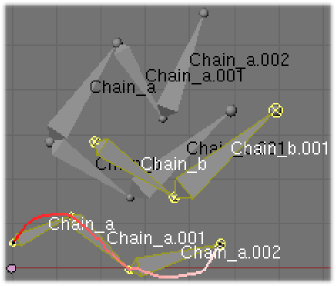
This is easier to understand with some examples.

In the following image, `armature.002` is set as the template, and the stroke maps with `chain_a` of this template. None of the bones are selected in the template. Note that there is no second stroke to map with chain `chain_b` of the template. The result is shown at right: Blender creates a copy of `chain_a` and matches the bones with the stroke.

Blender also creates a copy of `chain_b`, but this chain is not altered in any way; because this command can map only one selected chain with a stroke.


Skeleton Sketching
Quick Sketching
Overdraw Sketching
Stroke convers...: Retarget
Template: Armature.002
Retarget roll m...: View
Autoname Bones
Number:
Side:
Convert to Bones
Delete Strokes

Before conversion.



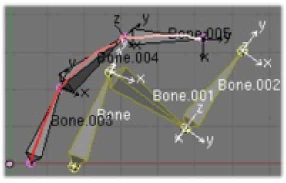
After conversion.

Roll methods

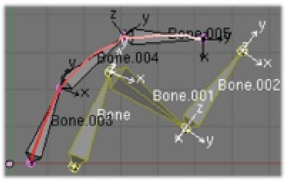
Now, let us see the settings of this bone roll method:

None, View, Joint buttons
: These three toggle buttons (mutually exclusive) control how the roll angle of newly created bones is affected:

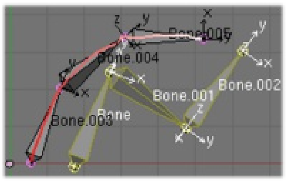
- No: Do not alter the bones roll (i.e. the new bones' rolls fit their reference ones).
- View: Roll each bone so that one of its X, Y or Z local axis is aligned (as much as possible) with the current view's Z axis.
- Joint: New bones roll fit their original rotation (as No option), but with regards to the bend of the joint with its parent.



With No roll option.



With View roll option.



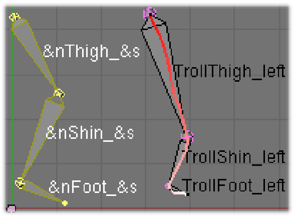
With Joint roll option.

Templating: bone roll example. The `Bone.003`-to-`Bone.005` chain is the mapped-to-stroke version of `Bone`-to-`Bone.002` selected one, and `Bone.001` has a modified roll angle.

Autonaming settings

Number and Side
: For these options must be enable the Autoname button. They control how the new bones are named. By default, they just take the same names as the originals from the template – except for the final number, increased as needed. However, if the template bones have "`&s`" somewhere in their name, this "placeholder" will be replaced in the "templated" bones' names by the content of the Side text field. Similarly, a "`&n`" placeholder will be replaced by the Number field content.

Finally, the Number field content will be auto-generated, producing a number starting from nothing, and increased each time you press the Convert to Bones button, and the "`&s`" placeholder will be replaced by the Side field content (relative to the local X axis: "`r`" for negative X values, "`l`" for positive ones).

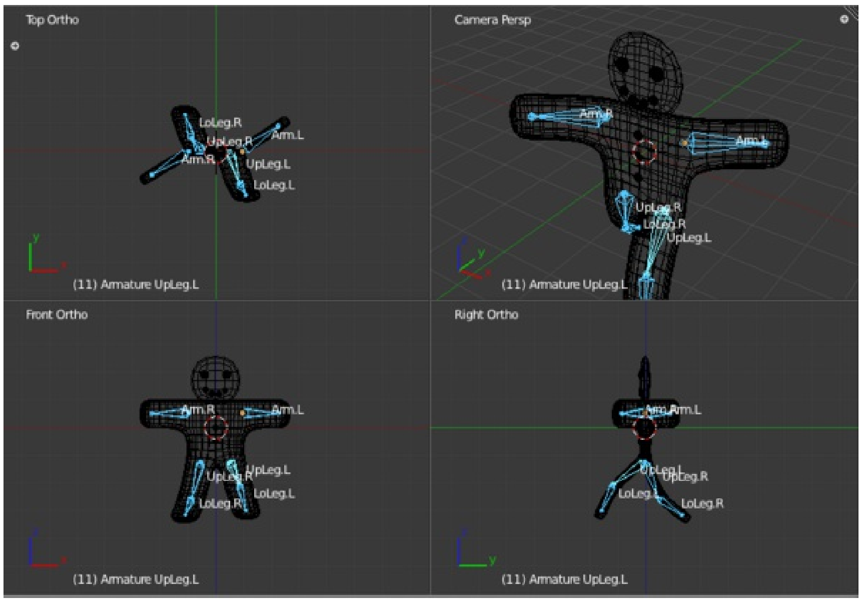


After conversion: the placeholders have been replaced by the content of the Side and Number text fields of the Skeleton Sketching panel.

Naming and placeholders, using a simple leg template.

Skinning

We have seen in previous pages how to design an armature, create chains of bones, etc. Now, having a good rig is not the final goal – unless you want to produce a “Dance Macabre” animation, you’ll likely want to put some flesh on your skeletons! Surprisingly, “linking” an armature to the object(s) it should transform and/or deform is called the “skinning” process…



The ginebread mesh skinned on its armature…

In Blender, you have two main skinning types:

- You can Parent/Constrain Objects to Bones – then, when you transform the bones in Pose mode, their “children” objects are also transformed, exactly as with a standard parent/children relationship… *The “children” are **never** deformed when using this method*.
- You can Using the Armature modifier on entire Mesh and then, adjust, some parts of this object to some bones inside this armature. This is the more complex and powerful method, and *the only way to really deform the geometry of the object*, i.e. to modify its vertices/control points relative positions.

Linking Objects to Bones

First step skining an armature by geometry is its parenting to bones. There are some settings necessary for an armature to deform a mesh:

- The meshobject needs to be parented to the armatureobject.
- The bones of the armature have to be adjusted their influence on parts of the mesh.

Object parenting to armature

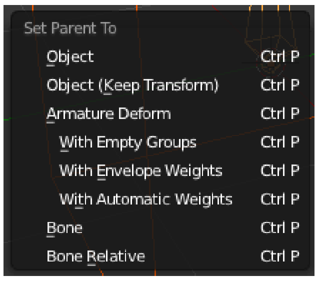
An object is parented to an armature by either:

- adding an Armature Modifier to the object and selecting the appropriate armature
- parenting the object to the armature (not as flexible but still working)

Object Parenting to Whole Armatures by Default Method

But before diving into this, let’s talk about the different ways to skin (parent) an object to a whole armature – as with **object skinning**, there is an “old parenting” method and a new, more flexible and powerful one, based on modifiers – which allows creation of very complex setups, with objects deformed by several armatures…

For *meshes and lattices only*, you can use the CtrlP parent shortcut in the 3D views (after having selected first the “skin” object, then the armature). The Make Parent To menu pops up, select the Armature entry. If the skinning object is a lattice, you're done; no more options are available. But with a child mesh, another Create Vertex Groups? menu appears, with the following options – all regarding the “vertex groups” skinning method:



Set Parent menu

With Empty Groups
: will create, if they don’t already exist, empty groups, one for each bone in the skinned armature, with these bones’ names. Choose this option if you have already created (and weighted…) all the vertex groups the mesh requires.

With Envelope Weights
: will create, as with Name Groups option, the needed vertex groups. However, it will also weight them according to the bones’ envelope settings (i.e. it will assign to each groups the vertices that are inside its bone’s influence area, weighted depending on their distance to this bone). This means that if you had defined vertex groups using same names as skinned bones, their content will be completely overridden! *You’ll get the same behavior as if you used the envelopes skinning method, but with vertex groups…*

Automatic Weights
: Creates, as with Envelope Weights option, the needed vertex groups, with vertices assigned and weighted using the newer “bone heat” algorithm.

Tuning Bone Influences on Parts of Geometry

The adjustins transformations of the mesh by the changes of the armature bones can either be done by:

- using the bone envelopes.
- editing vertex groups for each bone.

The first option is default and should work automatically.

The second option has to be set for each bone and vertexgroup individually, even though those setting can be mirrored in symmertrical meshes. For the second option to work the vertexgroup *must* have the same name as the bone that controls it.

Skinning to Objects' Shapes

We saw in the previous page how to link (parent) whole objects to armature bones – a way to control the transform properties of this object via a rig. However, armatures are much more powerful: they can deform the *shape* of an object (i.e. affect its ObData datablock – its vertices or control points…).

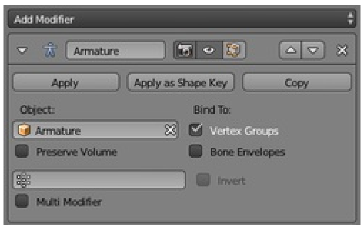
In this case, the child object is parented (skinned) to the whole armature, so that each of its bones controls a part of the "skin" object's geometry. This type of skinning is available for meshes, lattices, curves, surfaces, and texts (with more options for the first two types).

Bones can affect the object's shape in two ways:

- The "envelopes" process is available for all type of skinnable objects – it uses the "proximity" and "influence" of the bones to determine which part of the object they can deform.
- The "vertex groups" method is (obviously) reserved to meshes and lattices – one bone only affect the vertices in the group having the same name, using vertices' weights as influence value. A much more precise method, but also generally longer to set up.

Both methods have some common options, and can be mixed together.

Parenting to Whole Armatures with Armature modifier


Add Modifier
Armature
Apply | Apply as Shape Key | Copy
Object: Armature
Bind To: [✓] Vertex Groups; [ ] Bone Envelopes
[ ] Preserve Volume
[ ] Invert
[ ] Multi Modifier

The Armature modifier.

This "parenting" method will create an Armature modifier in the skinning object's modifiers stack. And so, of course, adding an Armature modifier to an object is the second, new skinning method (which also works for curves/surfaces/texts…). Follow the above link to read more about this modifier's specific options. Note that there is a way with new Armature modifiers to automatically create vertex groups and weight them; see the vertex groups method description below.

A single object can have several Armature modifiers (with e.g. different armatures, or different settings…), *working on top of each other,* ***or*** *mixing their respective effects* (depending whether their MultiModifier option is set, see their description for more details), and only one "virtual old parenting" one, which will always be at the top of the stack.

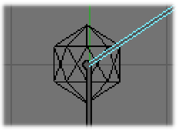
Note finally that for settings that are present in both the armature's Armature panel and in the objects' Armature modifier panel (namely, Vertex Groups/VertGroups, Envelopes, Preserve Volume and B-Bone Rest), *the modifier ones always override the armature ones*. This means that if, for example, you only enable the Envelopes deformation method of the armature, and then skin it with an object using an Armature modifier, where only VertGroups is enabled, *the object will only be deformed based on its "bones" vertex groups*, ignoring completely the bones' envelopes.

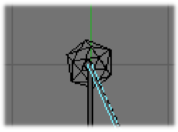
Common Options

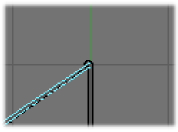
There are two armature-global skinning options that are common to both envelopes and vertex groups methods:

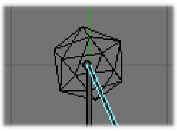
Preserve Volume (Armature modifier)
: This affects the way geometry is deformed, especially at bones' joints, when rotating them.
Without Preserve Volume, rotations at joints tend to scale down the neighboring geometry, up to nearly zero at **180°** from rest position.
With Preserve Volume, the geometry is no longer scaled down, but there is a "gap", a discontinuity when reaching **180°** from rest position.

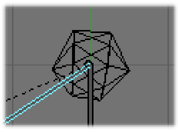
Example of Dual Quaternions skinning effects.

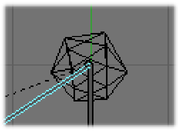

Initial state.


100° rotation, Preserve Volume disabled.


180° rotation, Preserve Volume disabled.

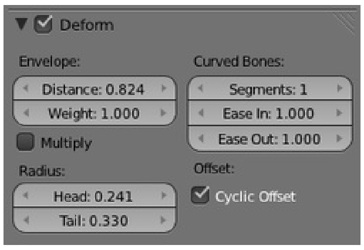

100° rotation, Preserve Volume enabled.


179.9° rotation, Preserve Volume enabled.


180.1° rotation, Preserve Volume enabled.

Note that the IcoSphere is deformed using the envelopes method.

Bone Deform Options



Deform

Envelope: Distance: 0.824, Weight: 1.000, Multiply

Radius: Head: 0.241, Tail: 0.330

Curved Bones: Segments: 1, Ease In: 1.000, Ease Out: 1.000

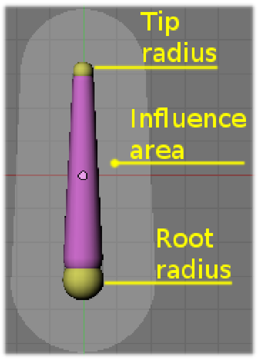
Offset: Cyclic Offset

Bone Deform Options

The bones also have some deforming options in their sub-panels (Armature Bones panel), that you can therefore define independently for each of them

Deform
: By disabling this setting (enabled by default), you can completely prevent a bone from deforming the geometry of the skin object.

Envelope



Bone influence areas for envelopes method.

Envelopes is the most general skinning method – it works with all available object types for skinning (meshes, lattices, curves, surfaces and texts). It is based on proximity between bones and their geometry, each bone having two different areas of influence, shown in the Envelope visualization:

- The inside area, materialized by the "solid" part of the bone, and controlled by both root and tip radius. Inside this zone, the geometry if fully affected by the bone.
- The outside area, materialized by the lighter part around the bone, and controlled by the Dist setting. Inside this zone, the geometry is less and less affected by the bone as it goes away – following a quadratic decay.

See the editing pages for how to edit these properties.

There is also a bone property, Weight (in each bone sub-panel, in Edit mode only, defaults to **1.0**), that controls the global influence of the bone over the deformed object, when using the envelopes method. It is only useful for the parts of geometry that are "shared", influenced by more than one bone (generally, at the joints…) – a bone with a high weight will have more influence on the result than one with a low weight… Note that when set to **0.0**, it has the same effect as disabling the Deform option.

Mult
: Short for 'Multiply'. This option controls how the two deforming methods interact when they are both enabled. By default, when they are both active, all vertices belonging to at least one vertex group are only deformed through the vertex groups method – the other "orphan" vertices being handled by the envelopes one.
When you enable this option, the "deformation influence" that this bone would have on a vertex (based from its envelope settings) is multiplied with this vertex's weight in the corresponding vertex group. In other words, the vertex groups method is further "weighted" by the envelopes method.

Radius
: Set the radius for the head and the tail of envelope bones.

Curved Bone

Curved Bones (previously known as B-bones) allow you make bones act like bezier curve segments, which results in smoother deformations for longer bones.

See the editing pages for how to edit these properties.

Vertex Groups

Vertex groups skinning method is only available for meshes and lattices – the only objects having vertex groups! Its principle is very simple: each bone only affects vertices belonging to a vertex group *having the same name as the bone*. So if you have e.g. a "`forearm`" bone, it will only affect the "`forearm`" vertex group of its skin object(s).

The influence of one bone on a given vertex is controlled by the weight of this vertex in the relevant group. Thus, the Weight Paint mode (Ctrl⇆ Tab with a mesh selected) is most useful here, to easily set/adjust the vertices' weights.

However, you have a few goodies when weight-painting a mesh already parented to (skinning) an armature. For these to work, you must:

- Select the armature.
- Switch to Pose mode (Ctrl⇆ Tab).
- Select the mesh to weight.
- Hit again Ctrl⇆ Tab to switch to Weight Paint mode.

Now, when you select a bone of the armature (which remained in Pose mode), you automatically activate the corresponding vertex group of the mesh – Very handy! Obviously, you can only select one bone at a time in this mode (so ⇧ Shift LMB clicking does not work).

This way, you can also apply to the active bone/vertex group one of the same "auto-weighting" methods as available when doing an "old-parenting" to armature (CtrlP):

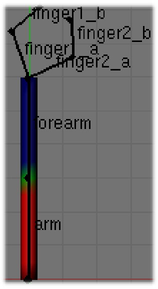
- Select the bone (and hence the vertex group) you want.
- Hit W, and in the Specials menu that pops up, choose either Apply Bone Envelopes to Vertex Groups or Apply Bone Heat Weights to Vertex Groups (names are self explanatory, I think…). Once again, even though these names are plural, you can only affect **one** vertex group's weights at a time with these options.

To automatically weight multiple bones, you can simply

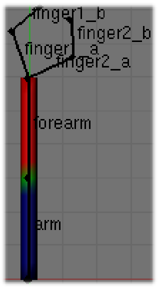
- Ctrl⇆ Tab out of Weight Paint Mode
- Select the Armature. It should be in Pose mode. If it isn't, go Ctrl⇆ Tab
- Select multiple bones ⇧ Shift LMB or hit 'a' (maybe twice).
- Select Mesh again
- If not in weight paint already, toggle back into Ctrl⇆ Tab
- Use the W menu to automatic weight. This will weight all the bones you selected in Pose Mode.

Obviously, the same vertex can belong to several groups, and hence be affected by several bones, with a fine tuning of each bone's influence using these vertex weights. Quite useful when you want to have a smooth joint. For example, when you skin an elbow, the upperarm vertex group contains the vertices of this part at full weight (**1.0**), and when reaching the elbow area, these weights decrease progressively to **0.0** when reaching the forearm zone – and vice versa for the forearm group weights… Of course, this is a very raw example – skinning a realistic joint is a big job, as you have to carefully find good weights for each vertex, to have the most realistic behavior when bending – and this is not an easy thing!

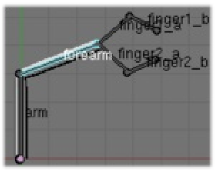
Example of vertex groups skinning method.



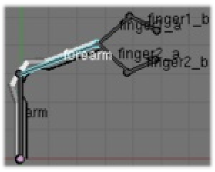
The weights of the `arm` vertex group.



The weights of the `forearm` vertex group.



The result when posing the armature.



The same pose, but using envelopes method rather that vertex groups.

See Also

Making good but short examples about skinning to shapes is not an easy thing – so if you want better examples, have a look to this BSoD tutorial, which illustrates (among many other things) the skinning of a simple human rig with a mesh object.

Retargeting Motion-capture Data to Skinned Rig

Retargeting — is a way to apply motion-capture data (acquired from real world) to a rig, so that it mimics the original movements realistically. This method also avoids laborious programming of each movement.

See more details here and here.

Posing

Once your armature is skinned by the needed object(s), you can start to pose it. Basically, by transforming its bones, you deform or transform the skin object(s). But you don't do that in Edit mode – remember that in this mode, you edit *the default, base, "rest" position of your armature*. You can't use the Object mode either, as here you can only transform whole objects…

So, armatures in Blender have a third mode, Pose, dedicated to this process. It's a sort of "object mode for bones". In rest position (as edited in Edit mode), each bone has its own position/rotation/scale to neutral values (i.e. **0.0** for position and rotation, and **1.0** for scale). Hence, when you edit a bone in Pose mode, you create an offset in its transform properties, from its rest position – this is quite similar to meshes' relative shape keys, in fact.

Posing Section Overview

In this section, we will see:

- The visualization features specific to Pose mode.
- How to select and edit bones in this mode.
- How to use pose library.
- How to use constraints to control your bones' DoF (degrees of freedom).
- How to use inverse kinematics features.
- How to use the Spline inverse kinematics features.

Even though it might be used for completely static purposes, posing is heavily connected with animation features and techniques.

In this part, we will try to focus on animation-independent posing, but this isn't always possible. So if you know nothing about animation in Blender, it might be a good idea to read the animation features and techniques chapter first, and then come back here.

See Also

As usual, see the tutorials for more demonstrative examples, and especially this BSoD one.

Visualization

We talk in this page about the armature visualization options available in all modes (the visualization types, the bones' shapes, etc.).

In Pose mode, you have extra features, colors and groups to help you visually categorize your bones, ghosts and motion paths to help you visualize armatures' animations.

Colors

In Pose mode, the bones can have different colors, following two different processes, controlled by the Color button (Armature panel, Editing context, F9):

- When it is disabled, bones are colored based on their "state" (i.e. if they use constraints, if they are posed, etc.).
- When it is enabled, bones are colored depending on which bone group they belong to (or as above if they belong to no group).

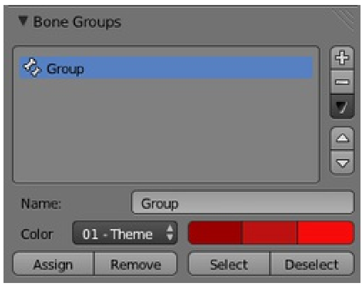
You can also mix both coloring methods, see below.

Coloring from Bone State

This is the default and oldest way – there are six different color codes, ordered here by precedence (i.e. the bone will be of the color of the topmost valid state):

- **Purple**: The Stride Root bone.
- **Orange**: A bone with a targetless Solver constraint.
- **Yellow**: A bone with an IK Solver constraint.
- **Green**: A bone with any other kind of constraint.
- **Blue**: A bone that is posed (i.e. has keyframes).
- **Gray**: Default state.

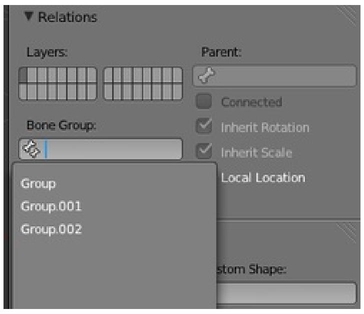
Coloring from Bone Group



The Bone Groups panel with a bone group (default colors).

The bone groups panel is available in the Object data editor for an armature. Bone groups facilitate the colouring (theming) of multiple bones. Bone groups are managed mostly in the Buttons window, Editing context (F9).

To create a new bone group, click on the Add Group button in the Bone Groups: buttons set (Link and Materials panel). Once created, you can use the top row of controls to select another group in the drop-down list ("arrows" button), rename the current group (text field), or delete it ("X" button).



The Bone Group drop-down list of a bone sub-panel.

To assign a selected bone to a given bone group you can do one of the following:

- In the Bone Groups, select an existing group, and click Assign
- In the Relations section of the Bones panel), use the Bone Group drop-down list to select the chosen one.

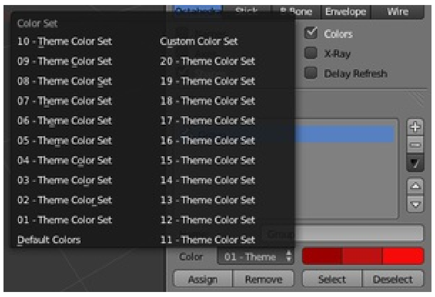
In the 3D views, using the Pose » Bone Groups menu entries, and/or the Bone Groups pop-up menu (CtrlG), you can:

Assign to New Group
    Assigns selected bones to a new bone group

Assign to Group
: Assigns selected bones to the selected Bone Groups

Remove Selected from Bone Groups
: Removes selected bones from all bone groups

Remove Bone Group
: Removes the active bone group



The Bone Color Set list of the bone group, and the color swatch of the chosen color theme.

You can also assign a “color theme” to a group (each bone will have these colors). Remember you have to enable the Colors button (Armature panel) to see these colors. Use the Bone Color Set drop-down list to select:

- The default (gray) colors (Default Colors).
- One of the twenty Blender presets (nn – Theme Color Set), common to all groups.
- A custom set of colors (Custom Set), which is specific to each group.

Below this list, you have three color swatches and a button.

- The first swatch is the color of unselected bones.
- The second swatch is the outline color of selected bones.
- The third swatch is the outline color of the active bone.

As soon as you click on a swatch (to change the color, through the standard color editing dialog), you are automatically switched to the Custom Set option.

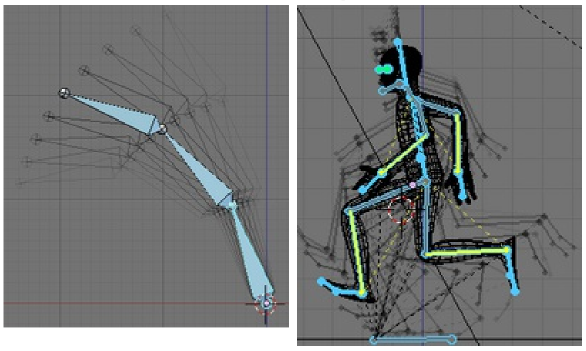
Ghosts

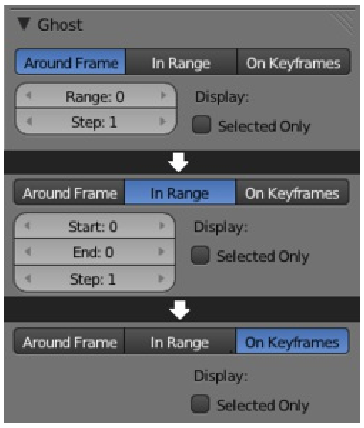
Mode: Pose mode

Panel: Visualisations

If you are a bit familiar with traditional cartoon creation, you might know that drawing artists use tracing paper heavily, to see several frames preceding the one they are working on. This allows them to visualize the overall movement of their character, without having to play it back… Well, Blender features something very similar for armatures in Pose mode: the “ghosts”.

Ghosts examples.





The Ghost panel showing the different options associated with different modes.

The ghosts are simply black drawings (more or less opaque) of the bones’ outlines as they are at certain frames.

The ghosts settings are found in the Visualisations panel (Editing context, F9), only available in Pose mode. You have three different types of ghosts, sharing more or less the same options:

Around Current Frame
: This will display a given number of ghosts before and after the current frame. The ghosts are shaded from opaque at the current frame, to transparent at the most distant frames. It has three options:

  Range
  : This numeric field specifies how many ghosts you'll have on both "sides" (i.e. a value of **5** will give you ten ghosts, five before the current frame, and five after).

  Step
  : This numeric field specifies whether you have a ghost for every frame (the default **1** value), or one each two frames, each three frames, etc.

  Selected Only
  : When enabled, you will only see the ghosts of selected bones (otherwise, every bone in the armatures has ghosts…)

In Range
: This will display the ghosts of the armature's bones inside a given range of frames. The ghosts are shaded from transparent for the first frame, to opaque at the last frame. It has four options:

  Start
  : This numeric field specifies the starting frame of the range (exclusive). Note that unfortunately, it cannot take a null or negative value – which means you can only see ghosts starting from frame **2** included…

  End
  : This numeric field specifies the ending frame of the range, and cannot take a value below GSta one.

  Step
  : Same as above.

On Keyframes
: This is very similar to the In Range option, but there are ghosts only for keyframes in the armature animation (i.e. frames at which you keyed one or more of the bones). So it has the same options as above, except for the GStep one (as only keyframes generate ghosts).
  Oddly, the shading of ghosts is reversed compared to In Range – from opaque for the first keyframe, to transparent for the last keyframe.

Finally, these ghosts are also active when playing the animation (AltA) – this is only useful with the Around Current Frame option, of course…

Note also that there is no "global switch" to disable this display feature – to do so, you have to either set Ghost to **0** (for Around Current Frame option), or the same frame number in both GSta and GEnd (for the two other ghosts types).

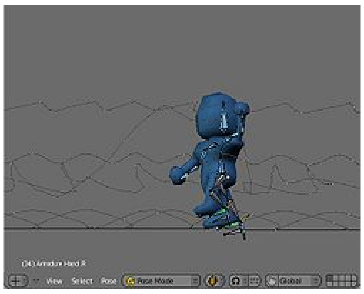
Motion Paths

Mode: Pose mode

Panel: Visualisations

Hotkey: WNum3, WNum4

Menu: Pose » Motion Paths » …


A motion paths example.

This feature allows you to visualize as curves the paths of bones' ends (either their tips, by default, or their roots).

Before we look at its options (all regrouped in the same Visualisations panel, in the Editing context, F9), let's first see how to display/hide these paths. Unlike ghosts, you have to do it manually – and you have to first select the bones you want to show/hide the motion paths. Then,

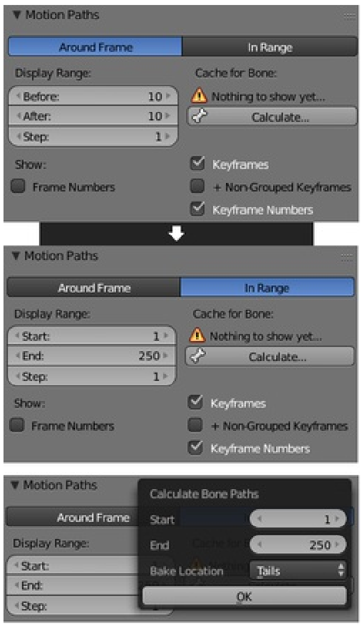
- To show the paths (or update them, if needed), click on the Calculate Path button of the Visualisations panel, or, in the 3D views,

select the Pose » Motion Paths » Calculate Paths menu entry (or use the Specials pop-up menu, WNum3).
- To hide the paths, click on the Clear Paths button, or, in the 3D views, do Pose » Motion Paths » Clear All Paths, or WNum4.

Remember: only selected bones and their paths are affected by these actions!

The paths are drawn in a light shade of gray for unselected bones, and a slightly blueish gray for selected ones. Each frame is materialized by a small white dot on the paths.

As with ghosts, the paths are automatically updated when you edit your poses/keyframes, and they are also active during animation playback (AltA, only useful when the Around Current Frame option is enabled).



The Motion Paths Panel showing options for the different modes

And now, the paths options:

Around Frame
: Around Frame, Display Paths of poses within a fixed number of frames around the current frame. When you enable this button, you get paths for a given number of frames before and after the current one (again, as with ghosts).;In Range
: In Range, Display Paths of poses within specified range.

Display Range
: Before/After

  Number of frames to show before and after the current frame (only for ‘Around Current Frame’ Onion-skinning method)

  Start/End

  Starting and Ending frame of range of paths to display/calculate (not for ‘Around Current Frame’ Onion-skinning method)

  Step

  This is the same as the GStep for ghosts – it allows you to only display on the path one frame for each *n* ones. Mostly useful when you enable the frame number display (see below), to avoid cluttering the 3D views.

Frame Numbers
: When enabled, a small number appears next to each frame dot on the path, which is of course the number of the corresponding frame…

Keyframes
: When enabled, big yellow square dots are drawn on motion paths, materializing the keyframes of their bones (i.e. only the paths of keyed bones at a given frame get a yellow dot at this frame).

Keyframe Nums
: When enabled, you’ll see the numbers of the displayed keyframes – so this option is obviously only valid when Show Keys is enabled.

+ Non-Grouped Keyframes
: For bone motion paths, search whole Action for keyframes instead of in group with matching name only (is slower)

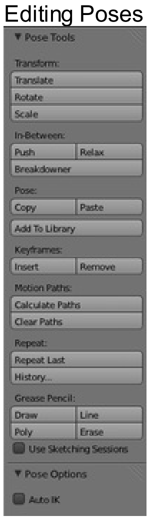
Calculate
: Start/End

  These are the start/end frames of the range in which motion paths are drawn. *You have to Calculate Paths again when you modify this setting*, to update the paths in the 3D views.
  Note that unlike with ghosts, the start frame is *inclusive* (i.e. if you set PSta to **1**, you’ll really see the frame **1** as starting point of the paths…).

Bake Location

By default, you get the tips' paths. By changing this setting to Tails, you'll get the paths of the bone's roots (remember that in Blender UI, bones' roots are called "heads"…). *You have to Calculate Paths again when you modify this setting*, to update the paths in the 3D views.

Editing Poses



Pose Tools

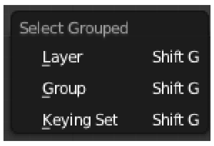
In Pose mode, bones behave like objects. So the transform actions (grab/rotate/scale, etc.) are very similar to the same ones in Object mode (all available ones are regrouped in the Pose » Transform sub-menu). However, there are some important specificities:

- Bones' relationships are crucial, as detailed below.
- The “transform center” of a given bone (i.e. its default pivot point, when it is the only selected one) is *its root*. Note by the way that some pivot point options seem to not work properly – in fact, except for the 3D Cursor one, all others appear to always use the median point of the selection (and not e.g. the active bone’s root when Active Object is selected, etc.).

Selecting Bones

Selection in Pose mode is very similar to the one in Edit mode, with a few specificities:

- You can only select *whole bones* in Pose mode, not roots/tips…

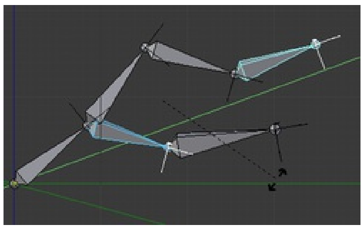


The Select Grouped pop-up menu.

- You can select bones based on their group and/or layer, through the Select Grouped pop-up menu (⇧ ShiftG):
  - To select all bones belonging to the same group(s) as the selected ones, use the In Same Group entry (⇧ ShiftGNum1).
  - To select all bones belonging to the same layer(s) as the selected ones, use the In Same Layer entry (⇧ ShiftGNum2).

Basic Posing

As previously noted, bones’ transformations are performed based on the *rest position* of the armature, which is its state as defined in Edit mode. This means that in rest position, in Pose mode, each bone has a scale of **1.0**, and null rotation and position (as you can see it in the Transform Properties panel, in the 3D views, N).



An example of locally-Y-axis locked rotation, with two bones selected. Note that the two green lines materializing the axes are centered on the armature’s center, and not each bone’s root…

Moreover, the local space for these actions is the bone's own one (visible when you enable the Axes option of the Armature panel). This is especially important when using axis locking – for example, there is no specific “bone roll” tool in Pose mode, as you can rotate around the bone’s main axis just by locking on the local Y axis (RYY)… This also works with several bones selected; each one is locked to its own local axis!

When you pose your armature, you are supposed to have one or more objects skinned on it! And obviously, when you transform a bone in Pose mode, its related objects or object’s shape is moved/deformed accordingly, in real time. Unfortunately, if you have a complex rig set-up and/or a heavy skin object, this might produce lag, and make interactive editing very painful. If you experience such troubles, try enabling the Delay Deform button of the Armature panel – the skin objects will only be updated once you validate the transform operation.

Auto IK

The auto IK option in the tool shelf enables a temporary ik constraint when posing bones. The chain acts from the tip of the selected bone to root of the uppermost parent bone. Note that this mode lacks options, and only works by applying the resulting transform to the bones in the chain.

Rest Pose

Once you have transformed some bones, if you want to return to their rest position, just clear their transformations (usual AltG/AltR/AltS shortcuts, or Pose » Clear Transform » Clear User Transform, WNum5, to clear everything at once… – commands also available in the Pose » Clear Transform sub-menu).

Note that in Envelope visualization, AltS does not clear the scale, but rather scales the Distance influence area of the selected bones (also available through the Pose » Scale Envelope Distance menu entry – only effective in Envelope visualization, even though it is always available…).

Conversely, you may define the current pose as the new rest position (i.e. “apply” current transformations to the Edit mode), using the Pose » Apply Pose as Restpose menu entry (or CtrlA and confirm the pop-up dialog). **When you do so, the skinned objects/geometry is also reset to its default, undeformed state**, which generally means you’ll have to skin it again.

Whereas in Edit mode, you always see your armature in its rest position, in Object and Pose ones, you see it by default in its *pose position* (i.e. as it was transformed in the Pose mode). If you want to see it in the rest position in all modes, enable the Rest Position button in the Armature panel (Editing context, F9).

In-Betweens

There are several tools for editing poses in an animation.

Relax Pose (Pose » In-Betweens » Relax Pose or AltE)
: Relax pose is somewhat related to the above topic – but it is only useful with keyframed bones (see the animation chapter). When you edit such a bone (and hence take it “away” from its “keyed position”), using this command will progressively “bring it back” to its “keyed position”, with smaller and smaller steps as it comes near it.

Push Pose (Pose » In-Betweens » Relax Pose or CtrlE)
: Push pose exaggerates the current pose.

Breakdowner (Pose » In-Betweens » Pose Breakdowner or ⇧ ShiftE)
: Creates a suitable breakdown pose on the current frame

There are also in Pose mode a bunch of armature-specific editing options/tools, like auto-bones naming, properties switching/enabling/disabling, etc., that we already described in the armature editing pages – follow the links above…

Copy/Paste Pose

Mode: Pose mode

Panel: 3D View header

Menu: Pose » Copy Current Pose, Pose » Paste Pose, Pose » Paste Flipped Pose

Copy and paste pose buttons in the 3D View header in Pose mode.

Blender allows you to copy and paste a pose, either through the Pose menu, or directly using the three “copy/paste” buttons found at the right part of the 3D views header:

Pose » Copy Current Pose
: to copy the current pose of selected bones into the pose buffer.

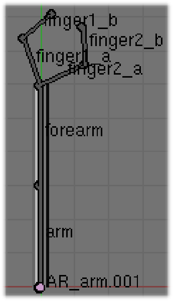
Pose » Paste Pose
: paste the buffered pose to the currently posed armature.

Pose » Paste Flipped Pose
: paste the **X axis mirrored** buffered pose to the currently posed armature.

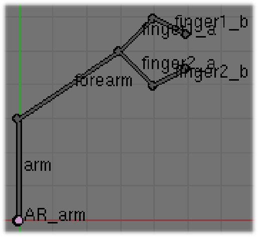
Here are important points:

- This tool works at the Blender session level, which means you can use it across armatures, scenes, and even files. However, the pose buffer is not saved, so you lose it when you close Blender.
- There is only one pose buffer.
- Only the selected bones are taken into account during copying (i.e. you copy only selected bones' pose).
- During pasting, on the other hand, bone selection has no importance. The copied pose is applied on a per-name basis (i.e. if you had a "`forearm`" bone selected when you copied the pose, the "`forearm`" bone of the current posed armature will get its pose when you paste it – and if there is no such named bone, nothing will happen…).
- What is copied and pasted is in fact the position/rotation/scale of each bone, in its own space. This means that the resulting pasted pose might be very different from the originally copied one, depending on:
  - The rest position of the bones, and
  - The current pose of their parents.

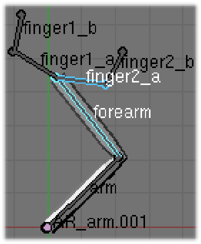
Examples of pose copy/paste.



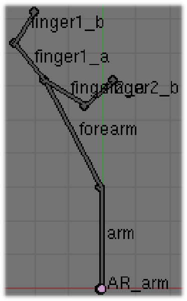
The rest position of our original armature.



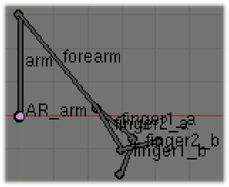
The rest position of our destination armature.



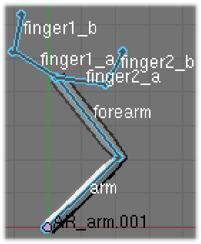
The first copied pose (note that only `forearm` and `finger2_a` are selected and hence copied)…



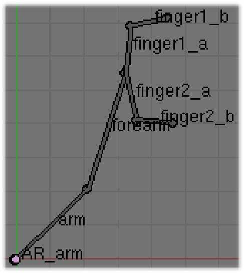
…pasted on the destination armature…



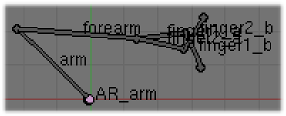
…and mirror-pasted on the destination armature.



The same pose as above is copied, but this time with all bones selected, …



…pasted on the destination armature…



…and mirror-pasted on the destination armature.

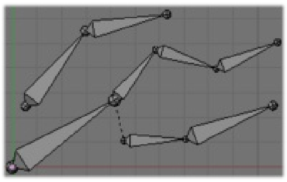
Effects of Bones Relationships

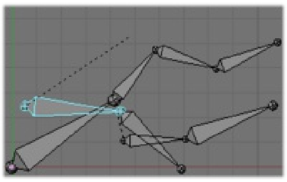
Bones relationships are crucial in Pose mode – they have important effects on transformations behavior.

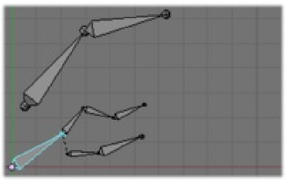
By default, children bones inherit:

- Their parent position, with their own offset of course.
- Their parent rotation (i.e. they keep a constant rotation relatively to their parent).
- Their parent scale, here again with their own offset.

Examples of transforming parented/connected bones.

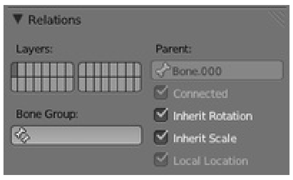

The armature in its rest


Rotation of a root bone.


Scaling of a root bone.

position.

Exactly like standard children objects. You can modify this behavior on a per-bone basis, using their sub-panels in the Armature Bones panel:



- [✓] Connected
- [✓] Inherit Rotation
- [✓] Inherit Scale
- [✓] Local Location

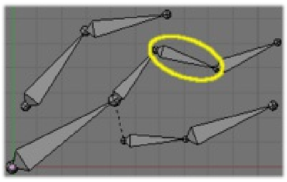
The Armature Bones panel in Pose mode.

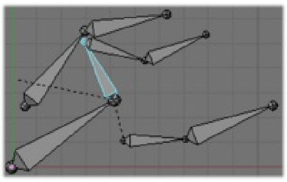
Inherit Rotation
: When disabled, this will “break” the rotation relationship to the bone’s parent. This means that the child will keep its rotation in the armature object space when its parent is rotated.

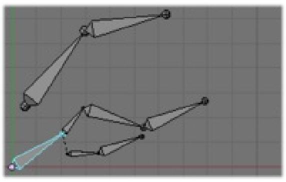
Inherit Scale
: When disabled, this will “break” the scale relationship to the bone’s parent.

These inheriting behaviors propagate along the bones’ hierarchy. So when you scale down a bone, all its descendants are by default scaled down accordingly. However, if you set one bone’s Inherit Scale or Inherit Rotation property on in this “family”, this will break the scaling propagation, i.e. this bone *and all its descendants* will no longer be affected when you scale one of its ancestors.

Examples of transforming parented/connected bones with Inherit Rotation disabled.


The yellow outlined Inherit Rotation disabled bone in the armature.

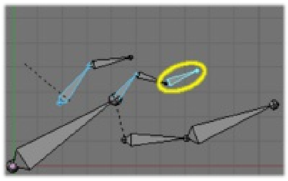

Rotation of a bone with a Inherit Rotation disabled bone among its descendants.


Scaling of a bone with a Inherit Rotation disabled bone among its descendants.

Connected bones have another specificity: they cannot be translated. Indeed, as their root must be at their parent’s tip, if you don’t move the parent, you cannot move the child’s root, but only its tip – which leads us to a child rotation. This is exactly what happens – when you hit G with a connected bone selected, Blender automatically switches to rotation operation.

Bones relationships also have important consequences on how selections of multiple bones behave when transformed. There are many different situations, so I’m not sure I list all possible ones below – but this should anyway give you a good idea of the problem:

- Non-related selected bones are transformed independently, as usual.


Scaling bones, some of them related.

- When several bones of the same “family” are selected, *only the “most parent” ones are really transformed* – the descendants are just handled through the parent relationship process, as if they were not selected (see *Scaling bones, some of them related* – the third tip bone, outlined in yellow, was only scaled down through the parent relationship, exactly as the unselected ones, even though it is selected and active. Otherwise, it should have been twice smaller!).
- When connected and unconnected bones are selected, and you start a grab operation, only the unconnected bones are affected.
- When a child connected hinge bone is in the selection, and the “most parent” selected one is connected, when you hit G, nothing happens – Blender remains in grab operation, which of course has no effect on a connected bone. This might be a bug, in fact, as I see no reason for this behavior…

So, when posing a chain of bones, you should always edit its elements from the root bone to the tip bone. This process is known as **forward kinematics**, or FK. We will see in a later page that Blender features another pose method, called **inverse kinematics**, or IK, which allows you to pose a whole chain just by moving its tip.

Note that this feature is somewhat extended/completed by the pose library tool.

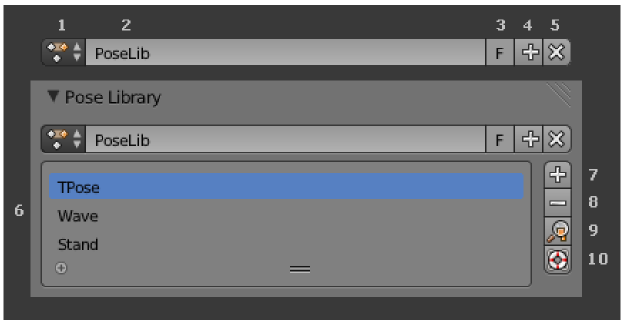
Pose Library

Intro

The *Pose Library* panel is used to save, apply, and manage different armature poses.

Pose Libraries are saved to *Actions*. They are not generally used as actions, but can be converted to and from.

Pose Library Panel



Properties > Armature > Pose Library.

1. Browse *Action / Pose Library* to be linked.

2. Name of the *Pose Library*.

3. Set Fake User.
   This will make blender save the *Pose Library* for if it has no users.

4. Add new *Pose Library* to the active object.

5. Remove the *Pose Library* from the active object.

6. A list of *Poses* for the active *Pose Library*.

7. Add Pose.
   Add New.

   Add a new *Pose* to the active *Pose Library* with the currect pose of the armature.

   Add New (Current Frame).

   *Add New*and *Replace Existing* automatically allocate a *Pose* to an *Action* frame.
   Add New(Currect Frame) will add a *Pose* to the *Pose Library* based on the currect frame of the *Time Cursor*.
   Its not a well supported feature.

   Replace Existing.

   Replace an existing *Pose* in the active *Pose Library* with the currect pose of the armature.

8. Remove the active *Pose* from the *Pose Library*.

9. Apply the active *Pose* to the selected *Pose Bones*.

10. Sanitize Action. Make *Action* suitable for use as a *Pose Library*.
    This is used to convert an *Action* to a *Pose Library*.
    A *Pose* is added to the *Pose Library* for each frame with keyframes.

Editing

3D View, Pose Mode.

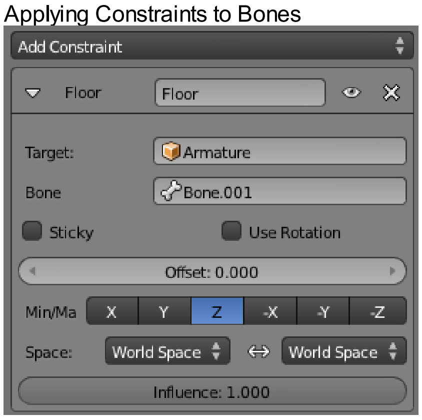
Browse Poses. CtrlL.

Add Pose. ⇧ ShiftL.

Rename Pose. ⇧ ShiftCtrlL.

Remove Pose. AltL.

Applying Constraints to Bones


Add Constraint

Floor: Floor

Target: Armature

Bone: Bone.001

Sticky [ ] Use Rotation [ ]

Offset: 0.000

Min/Ma: X Y [✓ Z] -X -Y -Z

Space: World Space ⇔ World Space

Influence: 1.000

The Constraints panel in Pose mode, with one Floor constraint applied to the active bone (Bone.001).

As bones behave like objects in Pose mode, they can also be constrained. This is why the Constraints panel is shown in both Object and Editing contexts in this mode… This panel contains the constraints *of the active bone* (its name is displayed at the top of the panel, in the To Bone:… static text field).

Constraining bones can be used to control their degree of freedom in their pose transformations, using e.g. the Limit constraints. You can also use constraints to make a bone track another object/bone (inside the same object, or in another armature), etc. And the inverse kinematics feature is also mainly available through the IK Solver constraint – which is specific to bones.

For example, a human elbow can't rotate backward (unless the character has broken his hand), nor to the sides, and its forward and roll rotations are limited in a given range (for example, depending on the rest position of your elbow, it may be from **0°** to **160°**, OR from **-45°** to **135°**). So you should apply a Limit Rotation constraint to the forearm bone (as the elbow movement is the result of rotating the forearm bone around its root).

Using bones in constraints, either as owners or as targets, is discussed in detail in the constraints pages.

Inverse Kinematics

IK simplifies the animation process, and makes it possible to make more advanced animations with lesser effort.

IK allows you to position the last bone in a bone chain and the other bones are positioned automatically. This is like how moving someone's finger would cause his arm to follow it. By normal posing techniques, you would have to start from the root bone, and set bones sequentially till you reach the tip bone: When each parent bone is moved, its child bone would inherit its location and rotation. Thus making tiny precise changes in poses becomes harder farther down the chain, as you may have to adjust all the parent bones first.

This effort is effectively avoided by use of IK.

Automatic IK

Automatic IK is a tool for quick posing, it can be enabled in the tool shelf in the 3D view, when in pose mode. When the Auto IK option is enabled, translating a bone will activate inverse kinematic and rotate bones higher up to follow the selected bone. By default, the length of the IK chain is as long as there are parent bones, and this length can be modified with ⇧ Shift PageUp, ⇧ Shift PageDown, or ⇧ Shift WheelUp, ⇧ Shift WheelDown.

This is a more limited feature than using an IK constraint, which can be configured, but it can be useful for quick posing.

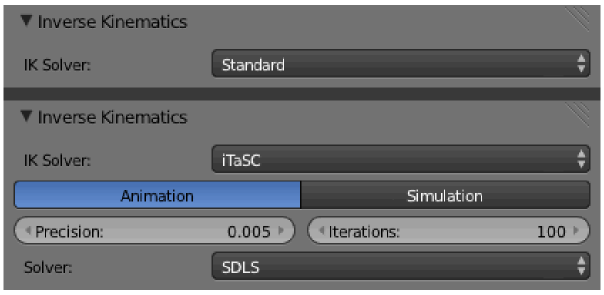
IK Contraints

IK is mostly done with bone constraints. They work by the same method but offer more choices and settings. Please refer to these pages for detail about the settings for the contraints:

- IK Solver
- Spline IK

Armature IK Panel

This panel is used to select the IK Solver type for the armature. *Standard* or *iTaSC*.

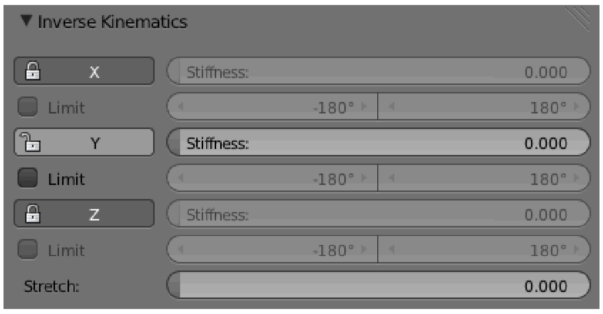

▼ Inverse Kinematics
IK Solver: Standard
▼ Inverse Kinematics
IK Solver: iTaSC
Animation | Simulation
Precision: 0.005 | Iterations: 100
Solver: SDLS

Properties > Armature > Inverse Kinematics Panel.

Most the time people will use the *Standard* IK solver.
There is some documentation for the *iTaSC* "instantaneous Task Specification using Constraints" IK solver here.
Robot IK Solver

Bone IK Panel

This panel is used to control how the *Pose Bones* work in the IK chain.


▼ Inverse Kinematics
X | Stiffness: 0.000
Limit | -180° | 180°
Y | Stiffness: 0.000
Limit | -180° | 180°
Z | Stiffness: 0.000
Limit | -180° | 180°
Stretch: 0.000

Properties > Bone > Inverse Kinematics Panel.

Lock
: Disallow movement around the axis.

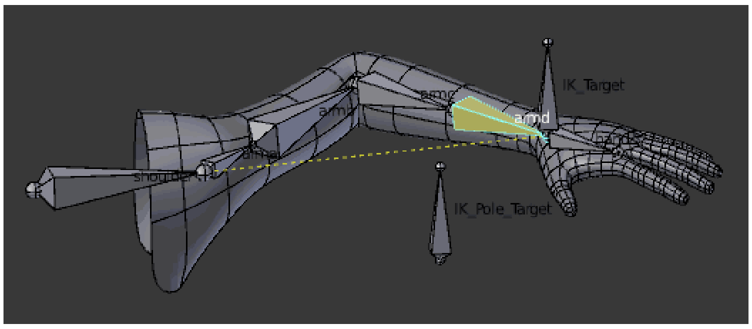
Stiffness
: Stiffness around the axis. Influence disabled if using *Lock*.

Limit
: Limit movement around the axis, specifide by the sliders.

Stretch
: Stretch influence to IK target. 0.000 is the same as disabled.

Arm Rig Example

This arm uses two bones to overcome the twist problem for the forearm. IK locking is used to stop the forearm from bending, but the forearm can still be twisted manually by pressing RYY in *Pose Mode*, or by using other constraints.



IK Arm Example.

File:IK Arm Example.blend

Note that, if a *Pole Target* is used, IK locking will not work on the root boot.

Spline IK

Spline IK is a constraint which aligns a chain of bones along a curve. By leveraging the ease and flexibility of achieving aesthetically pleasing shapes offered by curves and the predictability and well integrated control offered by bones, Spline IK is an invaluable tool in the riggers' toolbox. It is particularly well suited for rigging flexible body parts such as tails, tentacles, and spines, as well as inorganic items such as ropes.

Full description of the settings for the spline IK are detailed on the Spline IK page.

Basic Setup

The Spline IK Constraint is not strictly an 'Inverse Kinematics' method (i.e. IK Constraint), but rather a 'Forward Kinematics' method (i.e. normal bone posing). However, it still shares some characteristics of the IK Constraint, such as operating on multiple bones not being usable for Objects, and being evaluated after all other constraints have been evaluated. It should be noted that if a Standard IK chain and a Spline IK chain both affect a bone at the same time the Standard IK chain takes priority. Such setups are best avoided though, since the results may be difficult to control.

To setup Spline IK, it is necessary to have a chain of connected bones and a curve to constrain these bones to.

1. With the last bone in the chain selected, add a 'Spline IK' Constraint from the Bone Constraints tab in the Properties Editor.
2. Set the 'Chain Length' setting to the number of bones in the chain (starting from and including the selected bone) that should be influenced by the curve.
3. Finally, set the 'Target' field to the curve that should control the curve.

Congratulations, the bone chain is now controlled by the curve.

Settings and Controls

Roll Control

To control the 'twist' or 'roll' of the Spline IK chain, the standard methods of rotating the bones in the chain along their y-axes still apply. For example, simply rotate the bones in the chain around their y-axes to adjust the roll of the chain from that point onwards. Applying copy rotation constraints on the bones should also work.

Offset Controls

The entire bone chain can be made to follow the shape of the curve while still being able to be placed at an arbitrary point in 3D-space when the 'Chain Offset' option is enabled. By default, this option is not enabled, and the bones will be made to follow the curve in its untransformed position.

Thickness Controls

The thickness of the bones in the chain is controlled using the constraint's 'XZ Scale Mode' setting. This setting determines the method used for determining the scaling on the X and Z axes of each bone in the chain.

The available modes are:

- None - this option keeps the X and Z scaling factors as 1.0
- Volume Preserve - the X and Z scaling factors are taken as the inverse of the Y scaling factor (length of the bone), maintaining the 'volume' of the bone
- Bone Original - this options just uses the X and Z scaling factors the bone would have after being evaluated in the standard way

In addition to these modes, there is an option, 'Use Curve Radius'. When this option is enabled, the average radius of the radii of the points on the curve where the endpoints of each bone are placed, are used to derive X and Z scaling factors. This allows the scaling effects, determined using the modes above, to be tweaked as necessary for artistic control.

Tips for Nice Setups

- For optimal deformations, it is recommended that the bones are roughly the same length, and that they are not too long, to facilitate a better fit to the curve. Also, bones should ideally be created in a way that follows the shape of the curve in its 'rest pose' shape, to minimise the problems in areas where the curve has sharp bends which may be especially noticeable when stretching is disabled.
- For control of the curve, it is recommended that hooks (in particular, Bone Hooks, which are new in 2.5) are used to control the control-verts of the curve, with one hook per control-vert. In general, only a few control-verts should be needed for the curve (i.e. 1 for every 3-5 bones offers decent control).
- The type of curve used does not really matter, as long as a path can be extracted from it that could also be used by the Follow Path Constraint. This really depends on the level of control required from the hooks.
- When setting up the rigs, it is currently necessary to have the control bones (for controlling the curve) in a separate armature to those used for deforming the meshes (i.e. the deform rig containing the Spline IK chains). This is to avoid creating pseudo "Dependency Cycles", since Blender's Dependency Graph can only resolve the dependencies the control bones, curves, and Spline IK'ed bones on an object by object basis.

Constraints

Constraints are a way of connecting *transform properties* (position, rotation and scale) between objects. Constraints are in a way the object counterpart of the modifiers, which work on the object *data* (i.e. meshes, curves, etc.).

All constraints share a basic common interface, again with many similarities with the modifiers' one.

Use of Constraints

Even though constraints might be very useful in static scenes (as they can help to automatically position/rotate/scale objects), they were first designed for animation, as they allow you to limit/control the freedom of an object, either in absolute (i.e. in global space), or relatively to other objects.

Also note that constraints internally work using 4x4 transformation matrices only. When you use settings for specific rotation or scaling constraining, this information is being derived from the matrix only, not from settings in a Bone or Object. Especially for combining rotations with non-uniform or negative scaling this can lead to unpredictable behavior.

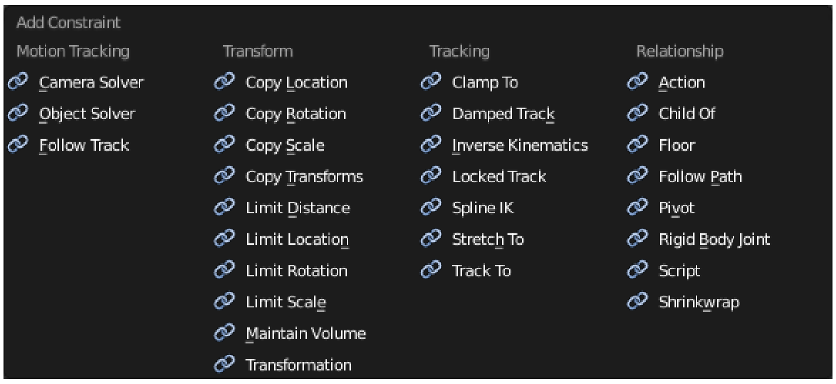
Constraining bones

Finally, there is a great rigging feature in Blender: in Pose mode, each bone of an armature behaves a bit like a standard object, and, as such, can be constrained. Most constraints work well with both objects and bones, but there are a few exceptions which are noted in the relevant constraints pages.

To learn more:

- Read about using constraints in object animation in the Animation chapter
- Read about using constraints in rigging in the Armatures

Available Constraints



The Constraint Menu

There are several types of constraints. We can classify them into four families:

- Motion Tracking
- Transform
- Tracking
- Relationship

There are constraints that works with their *owner* object and others that need a second object (the *target*) to work, sometimes of a specific type (e.g. a curve). In this case targeted constraints are shown as a dark blue dashed line drawn in the 3D view between the owner and target objects.

Motion Tracking

Camera Solver	This constraint is applied to a camera in the scene. Its constraints Blender camera movements according tracked and solved camera movements from a real filmed movie clip.
Object Solver	This constraint constraints object movements according tracked and solved object movements from a real filmed movie clip. Its applied to a object in the scene.
Follow Track	This constraint is applied to a camera in the scene. Its constraints Blender camera movements according tracked and solved camera movement by some path in a real filmed movie clip.

Transform Constraints

These constraints directly control/limit the transform properties of its owner, either absolutely or relatively in terms of its target properties.

Copy Location	Copies the location of the target (with an optional offset) to the owner, so that both move together.
Copy Rotation	Copies the rotation of the target (with an optional offset) to the owner, so that both rotate together.
Copy Scale	Copies the scale of the target (with an optional offset) to the owner, so that both scale together.
Copy	

Copy Transforms	Copies the transforms of the target to the owner, so that both transform together.
Limit Distance	Limits the position of the owner, so that it is nearer/further/exactly at the specified distance from the target.
Limit Location	Limits the owner's location inside a given range.
Limit Rotation	Limits the owner's rotation inside a given range.
Limit Scale	Limits the owner's scale inside a given range.
Transformation	Uses a property of the target (location, rotation or scale), to control a property (the same or a different one) of the owner.
Maintain Volume	Maintains the volume of a bone or an object.

Tracking Constraints

These constraints try, in various ways, to adjust their owner's properties so that it "points at" or "follows" the target.

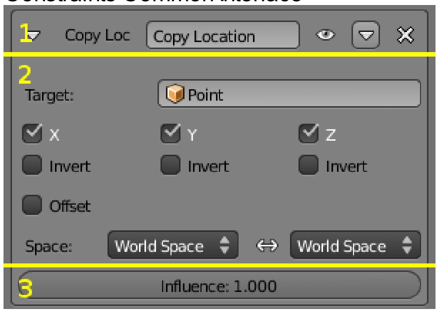
Clamp To	Clamps the owner to a given curve target.
Damped Track	Constrains one local axis of the owner to always point towards Target.
Inverse Kinematics	Bones only. Creates a chain of bones controlled by the target, using inverse kinematics.
Locked Track	The owner is tracked to the given target, but with a given axis' orientation locked.
Spline IK	Aligns a chain of bones along a curve.
Stretch To	Stretch the owner to the given target.
Track To	The owner is tracked to the given target.

Relationship Constraints

These are "misc" constraints.

Action	The owner executes an action, controlled by the target (driver).
Child Of	Allows a selective application of the effects of parenting to another object.
Floor	Uses the target's position (and optionally rotation) to define a "wall" or "floor" that the owner won't be able to cross.
Follow Path	The owner moves along the curve target.
Pivot	Allows the owner to rotate around a target object.
Rigid Body Joint	Creates a rigid joint (like a hinge…) between the owner and the "target" (child object).
Script	Uses a Python script as constraint.
Shrinkwrap	Limits the location of the owner at *the surface* (among other options) of the target.

Constraints Common Interface



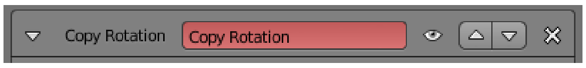
The three parts of a constraint interface

As with modifiers, an object (or bone, see the rigging chapter for details) can use several constraints at once. Hence, these constraints are organized in a stack which controls their order of evaluation (from top to bottom).

All constraints share a common basic interface, packed up in a sort of sub-panel, that is split into three parts:

1. The header, gathering most common settings.
2. The constraint's specific settings.
3. The influence and animation controls (the Rigid Body Joint constraints have no influence setting).

Constraints Header



A constraint header

The header of a constraint "sub-panel" is the same for all. From left to right, you have:

A small arrow
: This control allows you to show/hide the constraint's settings.

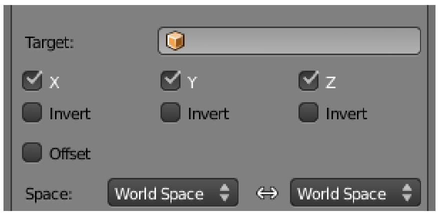
The constraint type
: This is just static text showing you what this constraint is…

The name field
: Here you can give your constraint a more meaningful name than the default one.
This control has another *important* purpose: it turns red when the constraint is not functional (as in *A constraint header*). As most constraints need a second "target" object to work (see below), when just added, they are in "red state", as Blender cannot guess which object or bone to use as target. This can also happen when you choose an invalid set of settings, e.g. with a Track To constraint of which the To and Up vectors are both set to the same axis.
As noted above, constraints in "red state" are ignored during the stack evaluation.

The "up"/"down" buttons
: As seen above, these allow you to move a constraint up/down in the stack.

The "X" control
: As seen above, this will delete (remove from the stack) the constraint.

Constraints Settings



The central part of a constraint's subpanel contains the constraint's settings, the target, and constraint space

The constraints settings area is of course specific to each constraint type. However, there are two points that are common to many constraints, so we will detail them here.

The target

Most constraints need another "target" object or bone to "guide" them. You select which by selecting its name in the Target field. Except for a few cases, you can use any type of object (camera, mesh, empty…); its object origin will be the target point.

When you type in the OB field a mesh or lattice name, a second Vertex Group field appears just below. If you leave it empty, the mesh or lattice will be used as a standard object target. But if you enter in this Vertex Group field the name of one of the mesh's or lattice's vertex groups, then the constraint will use the median point of this vertex group as target.

Similarly, if you type in the OB field an armature name, a second Bone field appears just below. If you enter in it the name of one of the armature's bones, then the constraint will use this bone's *root* as target. In some constraints, when you use a bone as target, another Head/Tail numeric field will also appear, that allows you to select where along the bone the target point will lay, from root (**0.0**) to tip (**1.0**) (remember that currently, in Blender UI, bones' roots are called "heads", and bones' tips, "tails"…).

The Constraint Space (Space)

For many constraints you can choose in which space it is evaluated/applied. In the Space drop-down lists, the right side one is the space that the owner is evaluated in (Owner Space). When such a constraint uses a target, you can also choose in which space the target is evaluated (Target Space). The Target Space drop-down list is on the left side. Both lists have the same options, depending on whether the element (owner or target) is a regular object, or a bone:

Local Space
: The object's properties are evaluated in its own local space, i.e. based on its rest position (without taking into account its parents' transformations in its chain, or its armature object's transformation).

Local With Parent (bones only)
: The bone properties are evaluated in its own local space, *including* the transformations due to a possible parent relationship (i.e. due to the chain's transformations above the bone).

Pose Space (bones only)
: The bone properties are evaluated in the armature object local space (i.e. independently from the armature transformations in Object mode). Hence, if the armature object has null transformations, Pose Space will have the same effect as World Space…

Local (Without Parent) Space (objects only)
: The object properties are evaluated in its own local space, *without* the transformations due to a possible parent relationship.

World Space (default setting)
: Here the object's or bone's properties are evaluated in the global coordinate system. This is the easiest to understand and most natural behavior, as it always uses the "visual" transform properties (i.e. as you see them in the 3D views).

Understanding the Constraint Space effects is not really easy (unless you are a geometry genius…). The best thing to do is to experiment with their different combinations, using e.g. two empties (as they materialize clearly their axes), and a Copy Rotation constraint (as rotations are the most demonstrative transformations, to visualize the various spaces specificities…).

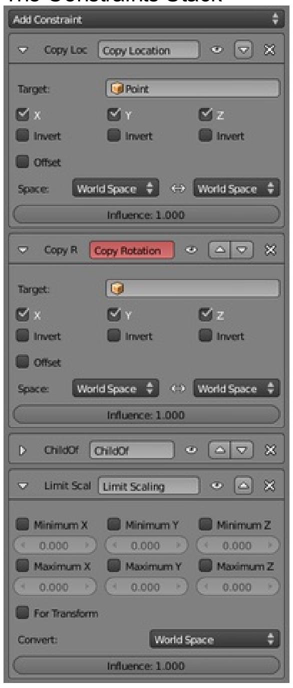
Influence

Influence

At the bottom of nearly all constraints, you have the Influence slider, which controls the influence of the constraint on its owner. As you might expect, **0.0** means that the constraint has no effect, and **1.0** means that the constraint has full effect. Using in-between values, you can have several constraints all working together on the same owner's properties. Note that if a constraint has a full influence on a given property, all other constraints above in the stack working on that same property will have no effect at all.

But the best thing with influence is that you can animate it with an Fcurve – see the constraints page of the animation chapter for more details about this.

The Constraints Stack



A constraint stack example

Constraints are evaluated from top to bottom of the constraint stack, shown in the Constraints panel.

1. In (*A constraint stack example*), first the location of the lamp is copied to the owner object.
2. The copy rotation constraint is ignored (red name, see below).
3. So the next constraint evaluated is the Child Of one, which is currently reduced.
4. Finally, the size of our cube is bounded by the Limit Scale last constraint.

   So here, the size of the cube is first controlled by the target of the Child Of constraint, within the limits allowed by the next Limit Scale constraint… As with modifiers, order is crucial!

You can move a constraint up and down the stack by using the small up/down arrow buttons that are drawn in its header, to the right of the constraint name. These buttons are only visible when needed, i.e. the top constraint has only the “down” button, the bottom constraint, only the “up” one – and when there is only one constraint in the stack, both buttons are hidden.

Adding/Removing a Constraint

To add a constraint, you can, in the Constraints panel, click on the… Add Constraint button! A menu shows up, listing all available constraints for the current active object (or bone in Pose mode (in which case the constraint will show up in the bone constraints menu). The new constraint is *always* added at the bottom of the stack.

You can also, in a 3D view, either:

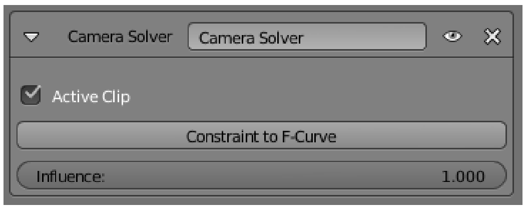
- Select only the future owner, hit Ctrl⇧ ShiftC, and in the Add Constraint to New Empty Object menu that pops up, select the constraint you want to add. If the chosen constraint needs it, a new Empty object will be automatically added as target, positioned at the owner’s center, and with null rotation.
- Select first the future target, and then the future owner, hit Ctrl⇧ ShiftC, and in the Add Constraint to Active Object (or Add Constraint to Active Bone) menu that pops up, select the constraint you want to add. If the chosen constraint needs it, the other selected object/bone will be used as target.

Note that these pop-up menus do not display all the available constraints.

To remove a constraint, click on the “X” button of the header of the constraint you want to delete, in the Constraints panel. You can also remove all constraints from the selected object(s), using the Object » Constraints » Clear Object Constraints (or Pose » Constraints » Clear Pose Constraints… or hit CtrlAltC).

Camera Solver Constraint

This constraint is applied to a camera in the scene. Its constraints Blender camera movements according tracked and solved camera movements from a real filmed movie clip.



Camera Solver panel

Options

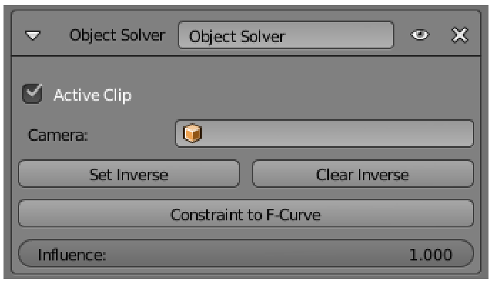
Active Clip
: Use active clip defined in scene

Constraint to F-Curve
: Create F-Curves for object which will copy object’s movement caused by this constraint

Influence
: Amount of influence constraint will have on the final solution

Object Solver Constraint

This constraint constraints object movements according tracked and solved object movements from a real filmed movie clip. Its applied to a object in the scene.



Object Solver panel

Options

Active Clip
: Use active clip defined in scene

Camera
: Camera to which motion is parented (if empty active scene camera is used)

Set Inverse
: Set inverse correction for ObjectSolver constraint

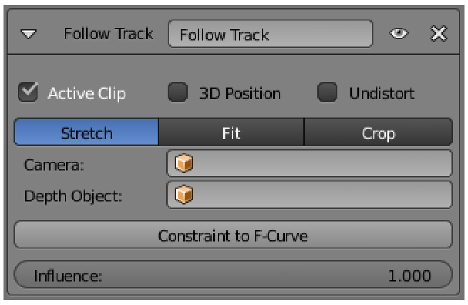
Clear Inverse
: Clear inverse correction for ObjectSolver constraint

Constraint to F-Curve
: Create F-Curves for object which will copy object's movement caused by this constraint

Influence
: Amount of influence constraint will have on the final solution

Follow Track Constraint

This constraint is applied to a camera in the scene. Its constraints Blender camera movements according tracked and solved camera movement by some path in a real filmed movie clip.



Follow Track panel

Options

Active Clip
: Use active clip defined in scene

3D Position
: Use 3D position of track to parent to

Undistort
: Parent to undistorted position of 2D track

Fit method footage in frame
: How the footage fits in the camera frame

  Stretch
  Fit
  Crop

Camera
: Camera to which motion is parented (if empty active scene camera is used)

Depth Object
: Object used to define depth in camera space by projecting onto surface of this object

Constraint to F-Curve
: Create F-Curves for object which will copy object’s movement caused by this constraint

Influence
: Amount of influence constraint will have on the final solution

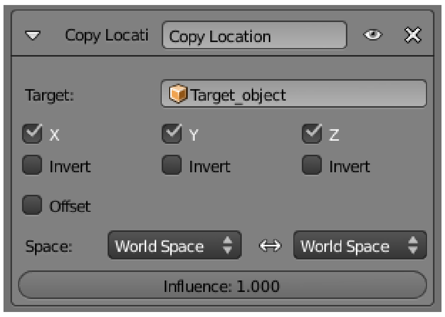
Copy Location Constraint

Description

The Copy Location constraint forces its owner to have the same location as its target.

Note that if you use such a constraint on a *connected* bone, it will have no effect, as it is the parent's tip which controls the position of your owner bone's root.

Options



Copy Location panel

Target
This constraint uses one target, and is not functional (red state) when it has none.

Bone
If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

Head/Tail
If a Bone is set as Target, a new field is displayed offering the optional choice of where along this bone the target point lies.

Vertex Group
If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.

X, Y, Z
These buttons control which axes (i.e. coordinates) are constrained – by default, all three ones are.

Invert
The Invert buttons invert their respective preceding coordinates.

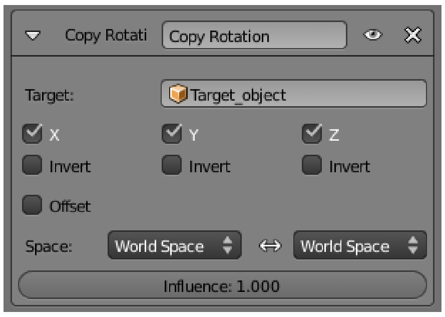
Offset
When enabled, this control allows the owner to be translated (using its current transform properties), relative to its target's position.

Space
This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

Copy Rotation Constraint

The Copy Rotation constraint forces its owner to match the rotation of its target.

Options


Copy Rotati Copy Rotation
Target: Target_object
✓ X ✓ Y ✓ Z
Invert Invert Invert
Offset
Space: World Space ↔ World Space
Influence: 1.000

Copy Rotation panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

  Bone
  : If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

    Head/Tail
    : If a Bone is set as Target, a new field is displayed offering the optional choice of where along this bone the target point lies.

  Vertex Group
  : If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.

X, Y, Z
: These buttons control which axes are constrained – by default, all three are on.

  Invert
  : The Invert buttons invert their respective rotation values.

Offset
: When enabled, this control allows the owner to be rotated (using its current transform properties), relative to its target's orientation.

Space
: This constraint allows you to choose in which space to evaluate its owner's and target's transform properties.

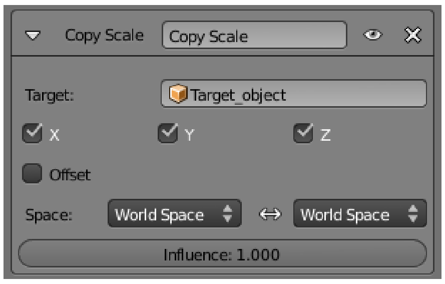
Copy Scale Constraint

Description

The Copy Scale constraint forces its owner to have the same scale as its target.

Here we talk of **scale**, not of **size**! Indeed, you can have two objects, one much bigger than the other, and yet both of them have the same scale. This is also true with bones: in Pose mode, they all have a unitary scale when they are in rest position, represented by their visible length.

Options


Copy Scale: Copy Scale
Target: Target_object
X ✓ Y ✓ Z ✓
Offset
Space: World Space ↔ World Space
Influence: 1.000

Copy Scale panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

  Bone
  : If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

    Head/Tail
    : If a Bone is set as Target, a new field is displayed offering the optional choice of where along this bone the target point lies.

  Vertex Group
  : If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.

X, Y, Z
: These buttons control along which axes the scale is constrained – by default, it is enabled along all three.

Offset
: When enabled, this control allows the owner to be scaled (using its current transform properties), relatively to its target’s scale.

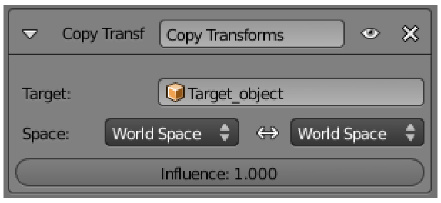
Space
: This constraint allows you to choose in which space to evaluate its owner’s and target’s transform properties.

Copy Transforms Constraint

Description

The Copy Transforms constraint forces its owner to have the same transforms as its target.

Options



Copy Transforms panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

    Bone
    : If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

        Head/Tail
        : If a Bone is set as Target, a new field is displayed offering the optional choice of where along this bone the target point lies.

    Vertex Group
    : If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.

Space
: This constraint allows you to choose in which space to evaluate its owner’s and target’s transform properties.

Limit Distance Constraint

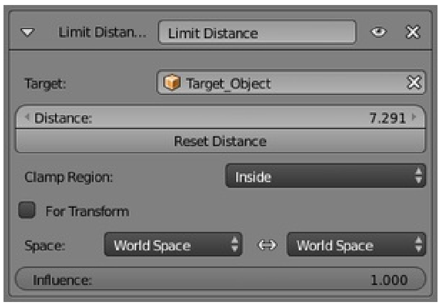
Description

The Limit Distance constraint forces its owner to stay either further from, nearer to, or exactly at a given distance from its target. In other words, the owner's location is constrained either outside, inside, or at the surface of a sphere centered on its target.

When you specify a (new) target, the Distance value is automatically set to correspond to the distance between the owner and this target.

Note that if you use such a constraint on a *connected* bone, it will have no effect, as it is the parent's tip which controls the position of your owner bone's root.

Options


Limit Distan... Limit Distance
Target: Target_Object
Distance: 7.291
Reset Distance
Clamp Region: Inside
For Transform
Space: World Space ⇔ World Space
Influence: 1.000

Limit Distance panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

  Bone
  : If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

    Head/Tail
    : If a Bone is set as Target, a new field is displayed offering the optional choice of where along this bone the target point lies.

  Vertex Group
  : If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.

Distance
: This numeric field sets the limit distance, i.e. the radius of the constraining sphere.

Reset Distance
: When clicked, this small button will reset the Distance value, so that it corresponds to the actual distance between the owner and its target (i.e. the distance before this constraint is applied).

Clamp Region
: The Limit Mode drop-down menu allows you to choose how to use the sphere defined by the Distance setting and target's center:

  Inside (default)
  : The owner is constrained *inside* the sphere.

  Outside
  : The owner is constrained *outside* the sphere.

  Surface
  : The owner is constrained *on the surface* of the sphere.

Limit Location Constraint

Description

An object or *unconnected* bone can be moved around the scene along the X, Y and Z axes. This constraint restricts the amount of allowed translations along each axis, through lower and upper bounds.

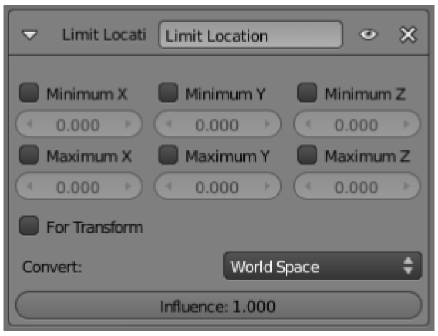
The limits for an object are calculated from its center, and the limits of a bone, from its root.

It is interesting to note that even though the constraint limits the visual and rendered location of its owner, its owner’s data block still allows (by default) the object or bone to have coordinates outside the minimum and maximum ranges. This can be seen in its *Transform Properties* panel (N). When an owner is grabbed and attempted to be moved outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally specified location.

Similarly, if its owner has an internal location that is beyond the limits, dragging it back into the limit area will appear to do nothing until the internal coordinates are back within the limit threshold (unless you enabled the *For Transform* option, see below).

Setting equal the min and max values of an axis, locks the owner’s movement along that axis… Although this is possible, using the *Transformation Properties* axis locking feature is probably easier!

Options



Limit Location panel

Minimum X, Minimum Y, Minimum Z
: These buttons enable the lower boundary for the location of the owner’s center along, respectively, the X, Y and Z axes of the chosen Space.
The numeric field below them controls the value of their limit.
Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Maximum X, Maximum Y, Maximum Z
: These buttons enable the upper boundary for the location of the owner’s center along, respectively, the X, Y and Z axes of the chosen Space.
Same options as above.

For Transform
: We saw that by default, even though visually constrained, the owner can still have coordinates out of bounds (as shown by the *Transform Properties* panel). Well, when you enable this button, this is no longer possible – the owner’s transform properties are also limited by the constraint.
Note however that the constraint does not directly modify the coordinates: you have to grab its owner one way or another for this to take effect…

Convert
: This constraint allows you to choose in which space to evaluate its owner’s transform properties.

Limit Rotation Constraint

Description

An object or bone can be rotated around the X, Y and Z axes. This constraint restricts the amount of allowed rotations around each axis, through lower and upper bounds.

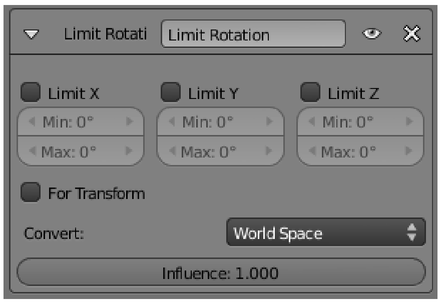
It is interesting to note that even though the constraint limits the visual and rendered rotations of its owner, its owner's data block still allows (by default) the object or bone to have rotation values outside the minimum and maximum ranges. This can be seen in the Transform Properties panel (N). When an owner is rotated and attempted to be rotated outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its rotation values will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally specified rotation.

Similarly, if its owner has an internal rotation that is beyond the limit, rotating it back into the limit area will appear to do nothing until the internal rotation values are back within the limit threshold (unless you enabled the For Transform option, see below).

Setting equal the min and max values of an axis, locks the owner's rotation around that axis… Although this is possible, using the Transformation Properties axis locking feature is probably easier.

This transform does not constrain the bone if it is manipulated by the IK solver. For constraining the rotation of a bone for IK purposes, see the "Inverse Kinematics" section of Bone properties.

Options



Limit Rotation panel

Limit X, LimitY, LimitZ
: These buttons enable the rotation limit around respectively the X, Y and Z axes of the owner, in the chosen Space.
The Min and Max numeric fields to their right control the value of their lower and upper boundaries, respectively.

  Note that:

  - The range of allowed limit values is clamped to +/- 180° this is a know limitation of the constraints system.
  - If a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.
  - Unlike the Limit Location constraint, you cannot enable separately lower or upper limits…

For Transform
: We saw that by default, even though visually constrained, the owner can still have rotations out of bounds (as shown by the Transform Properties panel). Well, when you enable this button, this is no more possible – the owner transform properties are also limited by the constraint.
Note however that the constraint does not directly modifies the rotation values: you have to rotate one way or the other its owner, for this to take effect…

Convert
: This constraint allows you to chose in which space evaluate its owner's transform properties.

Limit Scale Constraint

Description

An object or bone can be scaled along the X, Y and Z axes. This constraint restricts the amount of allowed scalings along each axis, through lower and upper bounds.

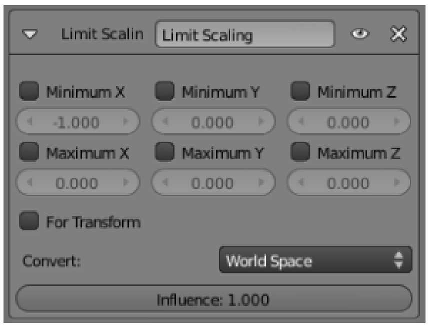
This constraint does not tolerate negative scale values (those you might use to mirror an object…): when you add it to an object or bone, even if no axis limit is enabled, nor the For Transform button, as soon as you scale your object, all negative scale values are instantaneously inverted to positive ones… And the boundary settings can only take strictly positive values.

It is interesting to note that even though the constraint limits the visual and rendered scale of its owner, its owner's data block still allows (by default) the object or bone to have scale values outside the minimum and maximum ranges (as long as they remain positive!). This can be seen in its Transform Properties panel (N). When an owner is scaled and attempted to be moved outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally-specified scale.

Similarly, if its owner has an internal scale that is beyond the limits, scaling it back into the limit area will appear to do nothing until the internal scale values are back within the limit threshold (unless you enabled the For Transform option, see below – or your owner has some negative scale values).

Setting equal the min and max values of an axis locks the owner's scaling along that axis. Although this is possible, using the Transformation Properties axis locking feature is probably easier.

Options



Limit Scale panel

Minimum/Maximum X, Y, Z
: These buttons enable the lower boundary for the scale of the owner along respectively the X, Y and Z axes of the chosen Space. The Min and Max numeric fields to their right control the value of their lower and upper boundaries, respectively.
Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

For Transform
: We saw that by default, even though visually constrained, and except for the negative values, the owner can still have scales out of bounds (as shown by the Transform Properties panel). Well, when you enable this button, this is no longer possible – the owner transform properties are also limited by the constraint.
Note however that the constraint does not directly modify the scale values: you have to scale its owner one way or another for this to take effect.

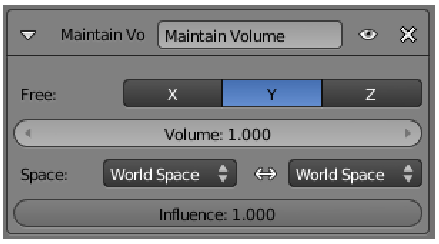
Convert
: This constraint allows you to choose in which space to evaluate its owner's transform properties.

Maintain Volume Constraint

Description

The Maintain Volume constraint limits the volume of a mesh or a bone to a given ratio of its original volume.

Option



Maintain Volume panel

Free X/Y/Z
: The free-scaling axis of the object.

Volume
: The bone's rest volume. Default is `1.0`.

Space
: This constraint allows you to choose in which space to evaluate its owner’s transform properties.

See also

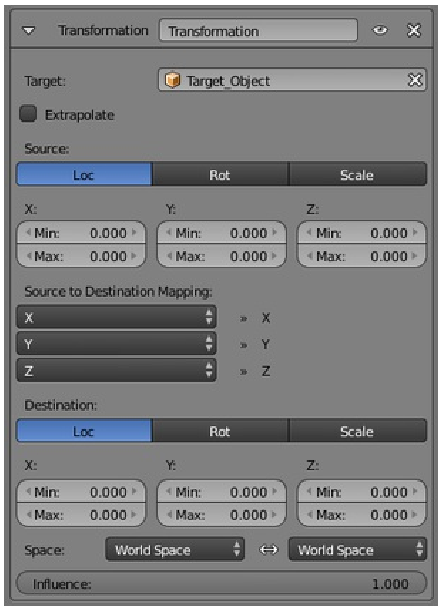
- Harkyman on the development of the Maintain Volume constraint, March 2010

Transformation Constraint

This constraint is more complex and versatile than the other “transform” constraints. It allows you to map one type of transform properties (i.e. location, rotation or scale) of the target, to the same or another type of transform properties of the owner, within a given range of values (which might be different for each target and owner property). You can also switch between axes, and use the range values not as limits, but rather as “markers” to define a mapping between input (target) and output (owner) values.

So, e.g. you can use the position of the target along the X axis to control the rotation of the owner around the Z axis, stating that **1 BU** along the target X axis corresponds to **10°** around the owner Z axis. Typical uses for this include gears (see note below), and rotation based on location setups.

Options



Transformation panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

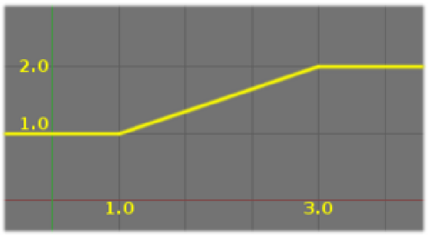
    Bone
    : If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

        Head/Tail
        : If a Bone is set as Target, a new field is displayed offering the optional choice of where along this bone the target point lies.

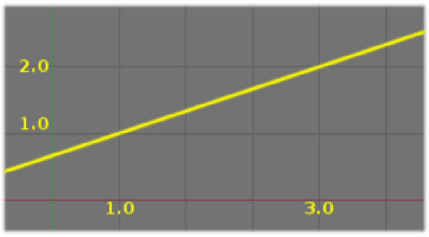
    Vertex Group
    : If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.

Extrapolate
: By default, the min and max values bound the input and output values; all values outside these ranges are clipped to them. When you enable this button, the min and max values are no longer strict limits, but rather “markers” defining a proportional (linear) mapping between input and corresponding output values.
Let’s illustrate that with two graphs (*The Extrapolate principles*). In these pictures, the input range (in abscissa) is set to [**1.0**, **4.0**], and its corresponding output range (in ordinate), to [**1.0**, **2.0**]. The yellow curve represents the mapping between input and output.

The Extrapolate principles.



Extrapolate disabled: the output values are bounded inside the [**1.0**, **2.0**] range.



Extrapolate enabled: the output values are “free” to proportionally follow the input ones.

Note that:

- When mapping transform properties to location (i.e. Loc Destination button is enabled), the owner’s existing location is added to the result of evaluating this constraint (exactly like when the Offset button of the Copy Location constraint is enabled…).
- Conversely, when mapping transform properties to rotation or scale, the owner’s existing rotation or scale is overridden by the result of evaluating this constraint.
- When using the rotation transform properties of the target as input, whatever the real values are, the constraint will always “take

them back” into the `[-180°, 180°[` range (e.g. if the target has a rotation of **420°** around its X axis, the values used as X input by the constraint will be `((420 + 180) modulo 360) - 180 = 60°`…). This is why this constraint is not really suited for gears!
- Similarly, when using the scale transform properties of the target as input, whatever the real values are, the constraint will always take their absolute values (i.e. invert negative ones).
- When a min value is higher than its corresponding max one, both are considered equal to the max one. This implies you cannot create “reversed” mappings…

Source
: It contains the input (from target) settings.

  The three Loc, Rot and Scale toggle buttons, mutually exclusive, allow you to select which type of property to use.
  The X:, Y: and Z: min and max numeric fields control the lower and upper bounds of the input value range, independently for each axis. Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Destination
: It contains the output (to owner) settings.

  - The three Loc, Rot and Scale toggle buttons, mutually exclusive, allow you to select which type of property to control.
  - The three Axis Mapping drop-down lists allow you to select which input axis to map to, respectively (from top to bottom), the X, Y and Z output (owner) axes.
  - The min and max numeric fields control the lower and upper bounds of the output value range, independently for each mapped axis. Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Space
: This constraint allows you to choose in which space to evaluate its owner’s and target’s transform properties.

Clamp To Constraint

The Clamp To constraint clamps an object to a curve. The Clamp To constraint is very similar to the Follow Path constraint, but instead of using the evaluation time of the target curve, Clamp To will get the actual location properties of its owner (those shown in the Transform Properties panel, N), and judge where to put it by “mapping” this location along the target curve.

One benefit is that when you are working with Clamp To, it is easier to see what your owner will be doing; since you are working in the 3D view, it will just be a lot more precise than sliding keys around on a time Ipo and playing the animation over and over.

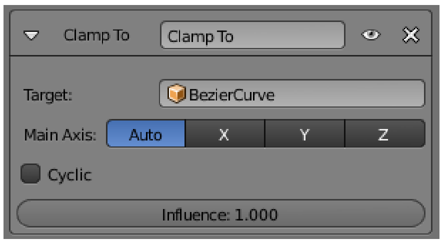
A downside is that unlike in the Follow Path constraint, Clamp To doesn’t have any option to track your owner’s rotation (pitch, roll, yaw) to the banking of the targeted curve, but you don’t always need rotation on, so in cases like this it’s usually a lot handier to fire up a Clamp To, and get the bits of rotation you do need some other way.

The mapping from the object's original position to its position on the curve is not perfect, but uses the following simplified algorithm (note, I am not the original code author so this may not be 100% accurate):

1. A "main axis" is chosen, either by the user, or as the longest axis of the curve's bounding box (the default).
2. The position of the object is compared to the bounding box of the curve in the direction of the main axis. So for example if X is the main axis, and the object is aligned with the curve bounding box's left side, the result is 0; if it is aligned with the right side, the result is 1.
3. If the cyclic option is unchecked, this value is clamped in the range 0-1.
4. This number is used as the curve time, to find the final position along the curve that the object is clamped to.

This algorithm does not produce exactly the desired result because curve time does not map exactly to the main axis position. For example an object directly in the centre of a curve will be clamped to a curve time of 0.5 regardless of the shape of the curve, because it is halfway along the curve's bounding box. However the 0.5 curve time position can actually be anywhere within the bounding box!

Options


Clamp To
Clamp To
Target: BezierCurve
Main Axis: Auto X Y Z
Cyclic
Influence: 1.000

Clamp To panel

Target
: The Target: field indicates which curve object the Clamp To constraint will track along. The Target: field must be a curve object type. If this field is not filled in then it will be highlighted in red indicating that this constraint does not have all the information it needs to carry out its task and will therefore be ignored on the constraint stack.

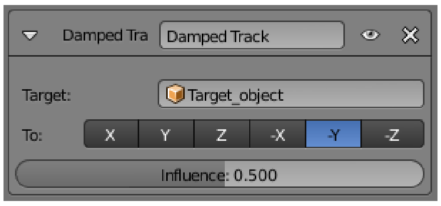
Main Axis
: This button group controls which global axis (X, Y or Z) is the main direction of the path. When clamping the object to the target curve, it will not be moved significantly on this axis. It may move a small amount on that axis because of the inexact way this constraint functions.
For example if you are animating a rocket launch, it will be the Z axis because the main direction of the launch path is up. The default Auto option chooses the axis which the curve is longest in (or X if they are equal). This is usually the best option.

Cyclic
: By default, once the object has reached one end of its target curve, it will be constrained there. When the Cyclic option is enabled, as soon as it reaches one end of the curve, it is instantaneously moved to its other end.
This is of course primarily designed for closed curves (circles & co), as this allows your owner to go around it over and over.

Damped Track Constraint

The Damped Track constraint constrains one local axis of the owner to always point towards Target. In another 3D software you can find it with the name "Look at" constraint.

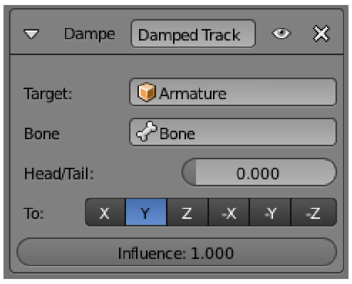
Options



Damped Track panel

Target (Mesh Object Type)
: This constraint uses one target, and is not functional (red state) when it has none.

Vertex Group
: If Target is a Mesh, a new field is displayed offering the optional choice to set a Vertex Group as target.



Damped Track for Bones

Target (Armature Object Type)

Bone
: If Target is an Armature, a new field is displayed offering the optional choice to set an individual bone as Target.

Head/Tail
: If Target is an Armature, a new field is displayed offering the optional choice to set whether the Head or Tail of a Bone will be pointed at by the Target. It is a slider value field which can have a value between 0 and 1. A value of 0 will point the Target at the Head/Root of a Bone while a value of 1 will point the Target at the Tail/Tip of a Bone.

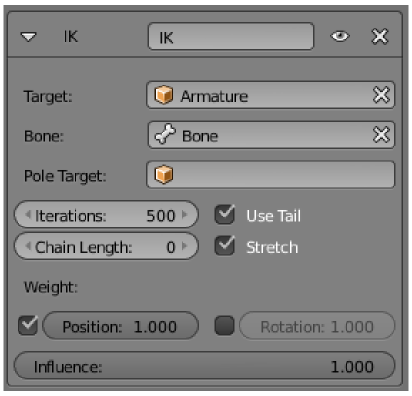
To
: Once the owner object has had a Damped Track constraint applied to it, you must then choose which axis of the object you want to point at the Target object. You can choose between 6 axis directions (-X, -Y, -Z, X, Y, Z). The negative axis direction cause the object to point away from the Target object along the selected axis direction.

IK Solver Constraint

The Inverse Kinematics constraint implements the *inverse kinematics* armature posing technique. Hence, it is only available for bones. To quickly create an IK constraint with a target, select a bone in pose mode, and press ⇧ Shift I.

This constraint is fully documented in the inverse kinematics page of the rigging chapter.

Options


IK IK
Target: Armature
Bone: Bone
Pole Target:
Iterations: 500 Use Tail
Chain Length: 0 Stretch
Weight:
Position: 1.000 Rotation: 1.000
Influence: 1.000

Inverse Kinematics panel

Target
: Must be an armature

Bone
: A bone in the armature

Pole Target
: Object for pole rotation

Iterations
: Maximum number of solving iterations

Chain Length
: How many bones are included in the IK effect. Set to 0 to include all bones

    Use Tail
    : Include bone's tail as last element in chain

    Stretch
    : Enable IK stretching

Weight

    Position
    : For Tree-IK: Weight of position control for this target

    Rotation
    : Chain follow rotation of target

Locked Track Constraint

The Locked Track constraint is a bit tricky to explain, both graphically and textually. Basically, it is a Track To constraint, but with a locked axis, i.e. an axis that cannot rotate (change its orientation). Hence, the owner can only track its target by rotating around this axis, and unless the target is in the plane perpendicular to the locked axis, and crossing the owner, this owner cannot really point at its target.

Let’s take the best real world equivalent: a compass. It can rotate to point in the general direction of its target (the magnetic North, or a neighbor magnet), but it can’t point *directly at it*, because it spins like a wheel on an axle. If a compass is sitting on a table and there is a magnet directly above it, the compass can’t point to it. If we move the magnet more to one side of the compass, it still can’t point *at* the target, but it can point in the general direction of the target, and still obey its restrictions of the axle.

When using a Locked Track constraint, you can think of the target as a magnet, and the owner as a compass. The Lock axis will function as the axle around which the owner spins, and the To axis will function as the compass’ needle. Which axis does what is up to you!

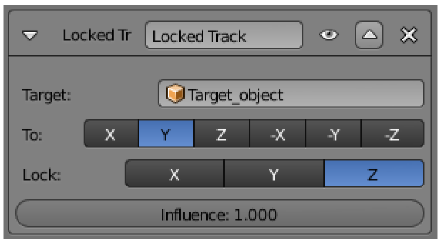
If you have trouble understanding the buttons of this constraint, read the tool-tips, they are pretty good. If you don’t know where your object’s axes are, turn on the Axis button in the Object menu's Draw panel. Or, if you’re working with bones, turn on the Axes button in the Armature menu's Display panel.

This constraint was designed to work cooperatively with the Track To constraint. If you set the axes buttons right for these two constraints, Track To can be used to point the axle at a primary target, and Locked Track can spin the owner around that axle to a secondary target.

This constraints also works very well for 2D billboarding.

This is all related to the topic discussed at length in the 2.49 BSoD tracking tutorial.

Options


Locked Tr: Locked Track
Target: Target_object
To: X [✓ Y] Z -X -Y -Z
Lock: X Y [✓ Z]
Influence: 1.000

Locked track panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

To
: The tracking local axis (Y by default), i.e. the owner’s axis to point at the target. The negative options force the relevant axis to point away from the target.

Lock
: The locked local axis (Z by default), i.e. the owner’s axis which cannot be re-oriented to track the target.

If you choose the same axis for *To* and *Lock*, the constraint will no longer be functional (red state).

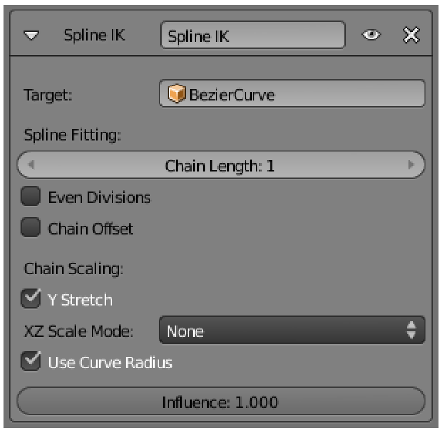
Spline IK Constraint

The Spline IK constraint aligns a chain of bones along a curve. By leveraging the ease and flexibility of achieving aesthetically pleasing shapes offered by curves and the predictability and well-integrated control offered by bones, Spline IK is an invaluable tool in the riggers' toolbox. It is particularly well suited for rigging flexible body parts such as tails, tentacles, and spines, as well as inorganic items such as ropes.

To set up Spline IK, it is necessary to have a chain of connected bones and a curve to constrain these bones to.

1. With the last bone in the chain selected, add a Spline IK constraint from the Bone Constraints tab in the Properties Editor.
2. Set the 'Chain Length' setting to the number of bones in the chain (starting from and including the selected bone) that should be influenced by the curve.
3. Finally, set Target to the curve that should control the curve.

Options


Spline IK   Spline IK
Target: BezierCurve
Spline Fitting:
Chain Length: 1
Even Divisions
Chain Offset
Chain Scaling:
✓ Y Stretch
XZ Scale Mode: None
✓ Use Curve Radius
Influence: 1.000

Spline IK panel

Target
: The target curve

Spline Fitting

- Chain Length
  : How many bones are included in the chain
- Even Division
  : Ignore the relative length of the bones when fitting to the curve
- Chain Offset
  : Offset the entire chain relative to the root joint

Chain Scaling

- Y stretch
  : Stretch the Y axis of the bones to fit the curve
- XZ Scale Mode
  - None
    : Don't scale the X and X axes (default)
  - Bone Original
    : Use the original scaling of the bones
  - Volume Preservation
    : Scale of the X and Z axes is the inverse of the Y scale
- Use Curve Radius
  : Average radius of the endpoints is used to tweak the X and Z scaling of the bones, on top of the X and Z scale mode

See also

This subject is seen in depth in the Rigging/Posing section.

- Blender.org 2.56 Release Log for Spline IK

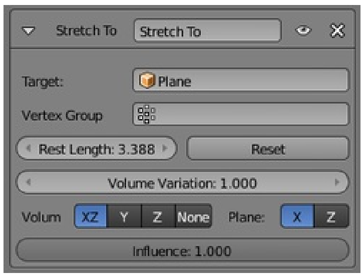
Stretch To Constraint

The Stretch To constraint causes its owner to rotate and scale its Y axis towards its target. So it has the same tracking behavior as the Track To constraint. However, it assumes that the Y axis will be the tracking and stretching axis, and doesn't give you the option of using a different one.

It also optionally has some raw volumetric features, so the owner can squash down as the target moves closer, or thin out as the target moves farther away. Note however that it is not the real volume of the owner which is thus preserved, but rather the virtual one defined by its scale values. Hence, this feature works even with non-volumetric objects, like empties, 2D meshes or surfaces, and curves.

With bones, the "volumetric" variation scales them along their own local axes (remember that the local Y axis of a bone is aligned with it, from root to tip).

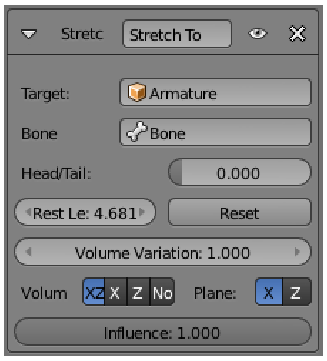
Options


Stretch To: Stretch To
Target: Plane
Vertex Group
Rest Length: 3.388
Reset
Volume Variation: 1.000
Volum: XZ Y Z None Plane: X Z
Influence: 1.000

Stretch To panel for a Mesh Object

Target (Mesh Object Type)
: This constraint uses one target, and is not functional (red state) when it has none.

    Vertex Group
    : When Target is a mesh, a new field is display where a vertex group can be selected.


Stretc: Stretch To
Target: Armature
Bone: Bone
Head/Tail: 0.000
Rest Le: 4.681
Reset
Volume Variation: 1.000
Volum: XZ X Z No Plane: X Z
Influence: 1.000

Stretch To panel for a Armature Object

Target (Armature Object Type)
: This constraint uses one target, and is not functional (red state) when it has none.

    Bone
    : When Target is an armature, a new field for a bone is displayed.

    Head/Tail
    : When using a Bone Target, you can choose where along this bone the target point lies.

Rest Length
: This numeric field sets the rest distance between the owner and its target, i.e. the distance at which there is no deformation (stretching) of the owner.

    Reset
    : When clicked, this small button will recalculate the Rest Length value, so that it corresponds to the actual distance between the owner and its target (i.e. the distance before this constraint is applied).

Volume Variation
: This numeric field controls the amount of "volume" variation proportionally to the stretching amount. Note that the **0.0** value is not allowed, if you want to disable the volume feature, use the None button (see below).

Volume
: These buttons control which of the X and/or Z axes should be affected (scaled up/down) to preserve the virtual volume while stretching along the Y axis.
If you enable the NONE button, the volumetric features are disabled.

Plane
: These buttons are equivalent to the *Up* ones of the Track To constraint: they control which of the X or Z axes should be maintained (as much as possible) aligned with the global Z axis, while tracking the target with the Y axis.

Track To Constraint

Description

The Track To constraint applies rotations to its owner, so that it always points a given “To” axis towards its target, with another “Up” axis permanently maintained as much aligned with the global Z axis (by default) as possible. This tracking is similar to the "billboard tracking" in 3D (see note below).

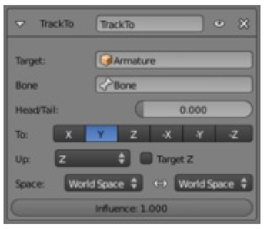
This is the preferred tracking constraint, as it has a more easily controlled constraining mechanism.

This constraint shares a close relationship to the Inverse Kinematics constraint in some ways. It is very important in rig design, and you should be sure to read and understand the 2.49 BSoD tracking tutorial, as it centers around the use of both of these constraints.

Billboard tracking

The term “billboard” has a specific meaning in real-time CG programming (i.e. video games!), where it is used for plane objects always facing the camera (they are indeed “trackers”, the camera being their “target”). Their main usage is as support for tree or mist textures: if they were not permanently facing the camera, you would often see your trees squeezing to nothing, or your mist turning into a millefeuille paste, which would be funny but not so credible.

Options



Track To panel

Targets
: This constraint uses one target, and is not functional (red state) when it has none.

  Bone
  : When Target is an armature, a new field for a bone is displayed.

  Head/Tail
  : When using a bone target, you can choose where along this bone the target point lies.

  Vertex Group
  : When Target is a mesh, a new field is display where a vertex group can be selected.

To
: The tracking local axis (Y by default), i.e. the owner’s axis to point at the target. The negative options force the relevant axis to point away from the target.

Up
: The “upward-most” local axis (Z by default), i.e. the owner’s axis to be aligned (as much as possible) with the global Z axis (or target Z axis, when the Target button is enabled).

Target Z
: By default, the owner’s Up axis is (as much as possible) aligned with the global Z axis, during the tracking rotations. When this button is enabled, the Up axis will be (as much as possible) aligned with the target’s local Z axis…

Space
: This constraint allows you to choose in which space to evaluate its owner’s and target’s transform properties.

If you choose the same axis for To and Up, the constraint will not be functional anymore (red state).

Action Constraint

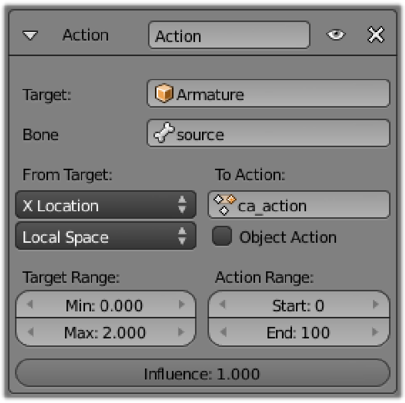
The Action constraint is powerful. It allows you control an Action using the transformations of another object.

The underlying idea of the Action constraint is very similar to the one behind the Drivers, except that the former uses a whole action (i.e. a bunch a Fcurves of the same type), while the latter controls a single Fcurve of their "owner"…

Note that even if the constraint accepts the Mesh action type, only the Object, Pose and Constraint types are really working, as constraints can only affect objects' or bones' transform properties, and not meshes' shapes… . Also note that only the object transformation (location, rotation, scale) is affected by the action, if the action contains keyframes for other properties they are ignored, as constraints do not influence those.

As an example, let's assume you have defined an Object action (it can be assigned to any object, or even no object at all), and have mapped it on your owner through an Action constraint, so that moving the target in the `[0.0, 2.0]` range along its X axis maps the action content on the owner in the `[0, 100]` frame range. This will mean that when the target's X property is **0.0**, the owner will be as if in frame **0** of the linked action; with the target's X property at **1.0**, the owner will be as if in frame **50** of the linked action, etc.

Options


Action: Action
Target: Armature
Bone: source
From Target: X Location, Local Space
To Action: ca_action
Object Action
Target Range: Min: 0.000, Max: 2.000
Action Range: Start: 0, End: 100
Influence: 1.000

Action panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

Bone
: When target is an armature object, use this field to select the target bone.

Transform Channel
: This drop-down list controls which transform property (location, rotation or scale along/around one of its axes) from the target to use as "action driver".

Target Space
: This constraint allows you to choose in which space to evaluate its target's transform properties.

To Action
: Select the name of the action you want to use.
: Even though it might not be in red state (UI refresh problems…), this constraint is obviously not functional when this field does not contain a valid action.

Object Action
: **Bones only**, when enabled, this option will make the constrained bone use the "object" part of the linked action, instead of the "same-named pose" part. This allows you to apply the action of an object to a bone.

Target Range Min/Max
: The lower and upper bounds of the driving transform property value.
: By default, both values are set to **0.0**
: Unfortunately, here again we find the constraints limitations:

  - When using a rotation property as "driver", these values are "mapped back" to the `[-180.0°, 180.0°[` range.
  - When using a scale property as "driver", these values are limited to null or positive values.

Action Range Start/End
: The starting and ending frames of the action to be mapped.
: Note that:

  - These values must be strictly positive.
  - By default, both values are set to **0**, which disables the mapping (i.e. the owner just gets the properties defined at frame **0** of the linked action…).

Notes

- When the linked action affects some location properties, the owner’s existing location is added to the result of evaluating this constraint (exactly as when the Offset button of the Copy Location constraint is enabled…).
- When the linked action affects some scale properties, the owner’s existing scale is multiplied with the result of evaluating this constraint.
- When the linked action affects some rotation properties, the owner’s existing rotation is overridden by the result of evaluating this constraint.
- Unlike usual, you can have a Start value higher than the End one, or a Min one higher than a Max one: this will reverse the mapping of the action (i.e. it will be “played” reversed…), unless you have both sets reversed, obviously!
- When using a Constraint action, it is the constraint *channel’s names* that are used to determine to which constraints of the owner apply the action. E.g. if you have a constraint channel named “trackto_empt1”, its keyed Influence and/or Head/Tail values (the only ones you can key) will be mapped to the ones of the owner’s constraint named “trackto_empt1”.
- Similarly, when using a Pose action (which is obviously only meaningful and working when constraining a bone!), it is the bone’s name that is used to determine which bone *channel’s names* from the action to use (e.g. if the constrained bone is named “arm”, it will use and only use the action’s bone channel named “arm”…). Unfortunately, using a Pose action on a whole armature object (to affect all the keyed bones in the action at once) won’t work…
- Note also that you can use the pose library feature to create/edit a Pose action datablock… just remember that in this situation, there’s one pose per frame!

Child Of Constraint

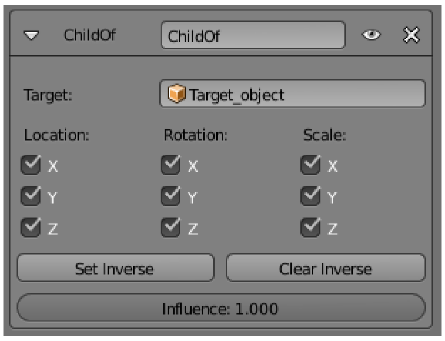
Child Of is the constraint version of the standard parent/children relationship between objects (the one established through the CtrlP shortcut, in the 3D views).

Parenting with a constraint has several advantages and enhancements, compared to the traditional method:

- You can have several different parents for the same object (weighting their respective influence with the Influence slider).
- As with any constraint, you can key (i.e. animate) its Influence setting. This allows the object which has a Child Of constraint upon it to change over time which target object will be considered the parent, and therefore have influence over the Child Of constrained object.

Don't confuse this "basic" object parenting with the one that defines the chains of bones inside of an armature. This constraint is used to parent an object to a bone (the so-called "object skinning"), or even bones to bones. But don't try to use it to define chains of bones.

Options



Child Of panel

Target
: The target object that this object will act as a child of. This constraint uses one target, and is not functional (red state) when it has none. If Target is an armature or a mesh, a new name field appears where a name of a Bone or a Vertex Group can be selected.

Location X, Y, Z
: Each of these buttons will make the parent affect or not affect the location along the corresponding axis.

Rotation X, Y, Z
: Each of these buttons will make the parent affect or not affect the rotation around the corresponding axis.

Scale X, Y, Z
: Each of these buttons will make the parent affect or not affect the scale along the corresponding axis.

Set Inverse
: By default, when you parent your owner to your target, the target becomes the origin of the owner's space. This means that the location, rotation and scale of the owner are offset by the same properties of the target. In other words, the owner is transformed when you parent it to your target.
This might not be desired! So, if you want to restore your owner to its before-parenting state, click on the Set Inverse button.

Clear Inverse
: This button reverses (cancels) the effects of the above one, restoring the owner/child to its default state regarding its target/parent.

Tips

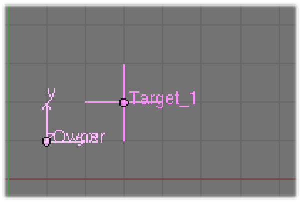
When creating a new parent relationship using this constraint, it is usually necessary to click on the Set Inverse button after assigning the parent. As noted above, this cancels out any unwanted transform from the parent, so that the owner returns to the location/rotation/scale it was in before the constraint was applied.

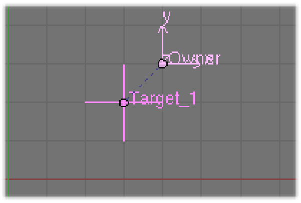
About the toggle buttons that control which target's (i.e. parent's) individual transform properties affect the owner, it is usually best to leave them all enabled, or to disable all three of the given Location, Rotation or Scale transforms.

Technical Note

If you use this constraint with all channels on, it will use a straight matrix multiplication for the parent relationship, not decomposing the parent matrix into loc/rot/size. This ensures any transformation correctly gets applied, also for combinations of rotated and non-uniform scaled parents.

Examples





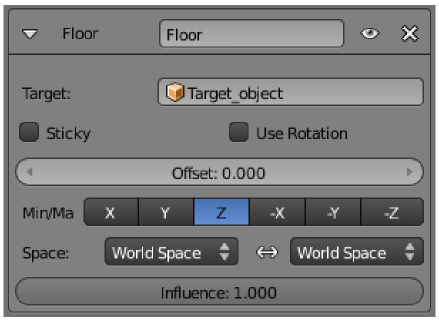
1. No constraint **2. Child Of just added**

Floor Constraint

Description

The Floor constraint allows you to use its target position (and optionally rotation) to specify a plane with a “forbidden side”, where the owner cannot go. This plane can have any orientation you like. In other words, it creates a floor (or a ceiling, or a wall)! Note that it is only capable of simulating entirely flat planes, even if you use the Vertex Group option. It cannot be used for uneven floors or walls.

Options



Floor panel

Targets
: This constraint uses one target, and is not functional (red state) when it has none.

  Bone
  : When Target is an armature, a new field for a bone is displayed.

  Vertex Group
  : When Target is a mesh, a new field is display where a vertex group can be selected.

Sticky
: This button makes the owner immovable when touching the “floor” plane (it cannot slide around on the surface of the plane any more). This is fantastic for making walk and run animations!

Use Rotation
: This button forces the constraint to take the target’s rotation into account. This allows you to have a “floor” plane of any orientation you like, not just the global XY, XZ and YZ ones…

Offset
: This numeric field allows you to offset the “floor” plane from the target’s center, by the given number of Blender Units. Use it e.g. to account for the distance from a foot bone to the surface of the foot’s mesh.

Max/Min
: This set of (mutually exclusive) buttons controls which plane will be the “floor”. The buttons’ names correspond indeed to the *normal* to this plane (e.g. enabling Z means “XY plane”, etc.)
By default, these normals are aligned with the *global* axes. However, if you enable Use Rotation (see above), they will be aligned with the *local target’s axes*.
As the constraint does not only define an uncrossable plane, but also a side of it which is forbidden to the owner, you can choose which side by enabling either the positive or negative normal axis… E.g, by default (Z), the owner is stuck in the positive Z coordinates.

Space
: This constraint allows you to choose in which space to evaluate its owner’s and target’s transform properties.

Follow Path Constraint

The Follow Path constraint places its owner onto a *curve* target object, and makes it move along this curve (or path). It can also affect its owner's rotation to follow the curve's bends, when the Follow Curve option is enabled.

The owner is always evaluated in the global (world) space:

- Its location (as shown in the Transform Properties panel, N) is used as an offset from its normal position on the path. E.g. if you have an owner with the `(1.0, 1.0, 0.0)` location, it will be one BU away from its normal position on the curve, along the X and Y axis. Hence, if you want your owner *on* its target path, clear its location (AltG)!
- This location offset is also proportionally affected by the *scale of the target curve*. Taking the same `(1.0, 1.0, 0.0)` offset as above, if the curve has a scale of `(2.0, 1.0, 1.0)`, the owner will be offset *two* BU along the X axis (and one along the Y one)…
- When the Curve Follow option is enabled, its rotation is also offset to the one given by the curve (i.e. if you want the Y axis of your object to be aligned with the curve's direction, it must be in rest, non-constrained state, aligned with the global Y axis). Here again, clearing your owner's rotation (AltR) might be useful…

The movement of the owner along the target curve/path may be controlled in two different ways:

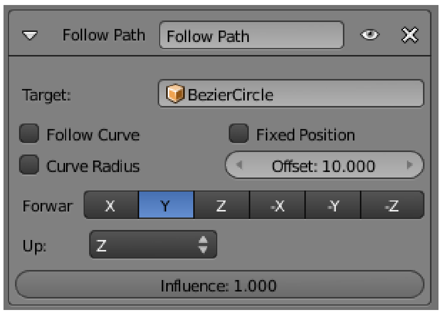
- The most simple is to define the number of frames of the movement, in the Path Animation panel of the Object Data context, via the numeric field Frames, and its start frame via the constraint's Offset option (by default, start frame: 1 [= offset of 0)], duration: 100).
- The second way, much more precise and powerful, is to define a Evaluation Time interpolation curve for the Target path (in the Graph Editor. See the animation chapter to learn more about Fcurves.
- If you don't want your owner to move along the path, you can give to the target curve a flat Speed FCurve (its value will control the position of the owner along the path).

Follow Path is another constraint that works well with the Locked Track one. One example is a flying camera on a path. To control the camera's roll angle, you can use a Locked Track and a target object to specify the up direction, as the camera flies along the path.

Follow Path and Clamp To
Do not confuse these two constraints. Both of them constraint the location of their owner along a curve, but Follow Path is an "animation-only" constraint, inasmuch that the position of the owner along the curve is determined by the time (i.e. current frame), whereas the Clamp To constraint determines the position of its owner along the curve using one of its location properties' values.

Note
Note that you also need to keyframe Evaluation Time for the Path. Select the path, go to the path properties, set the overall frame to the first frame of the path (e.g. frame 1), set the value of Evaluation time to the first frame of the path (e.g. 1), right click on Evaluation time, select create keyframe, set the overall frame to the last frame of the path (e.g. frame 100), set the value of Evaluation time to the last frame of the path (e.g. 100), right click on Evaluation time, select create keyframe.

Options



Follow Path panel

Target
: This constraint uses one target, which *must be a curve object*, and is not functional (red state) when it has none.

Curve Radius
: Objects scale by the curve radius. See Curve Editing

Fixed Position
: Object will stay locked to a single point somewhere along the length of the curve regardless of time

Offset
: The number of frames to offset from the "animation" defined by the path (by default, from frame **1**).

Follow Curve
: If this option is not activated, the owner's rotation isn't modified by the curve; otherwise, it's affected depending on the following options:

    Forward
    : The axis of the object that has to be aligned with the forward direction of the path (i.e. tangent to the curve at the owner's position).

    Up
    : The axis of the object that has to be aligned (as much as possible) with the world Z axis.
    In fact, with this option activated, the behavior of the owner shares some properties with the one caused by a Locked Track constraint, with the path as "axle", and the world Z axis as "magnet".

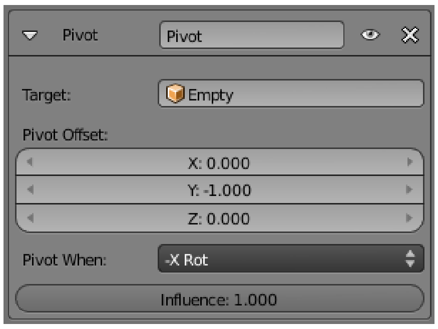
Pivot Constraint

Description

The Pivot constraint allows the owner to rotate around a target object.

It was originally intended for foot rigs.

Options



Pivot
Target: Empty
Pivot Offset:
X: 0.000
Y: -1.000
Z: 0.000
Pivot When: -X Rot
Influence: 1.000

Pivot panel

Target
: The object to be used as a pivot point

  Bone
  : When Target is an armature, a new field for a bone is displayed.

  Head/Tail
  : When using a bone target, you can choose where along this bone the target point lies.

  Vertex Group
  : When Target is a mesh, a new field is display where a vertex group can be selected.

Pivot Offset
: Offset of pivot from target

Pivot When
Always, Z Rot, Y Rot, ...

Example

[video link]

See also

- Blender Artists Forum: *Head-Tail pivot Constrain proposal (with Video and .blend)*, the thread where the constraint was first proposed

Rigid Body Joint Constraint

Description

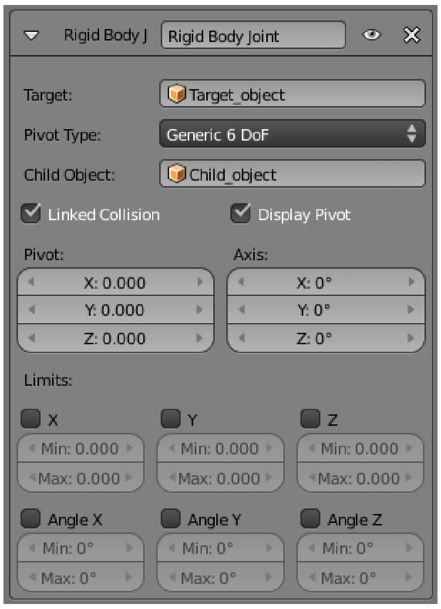
The Rigid Body Joint constraint is very special. Basically, it is used by the physical part of the Blender Game Engine to simulate a joint between its owner and its target. It offers four joint types: hinge type, ball-and-socket type, cone-twist, and generic six-DoF (degrees of freedom) type.

The joint point and axes are defined and fixed relative to the owner. The target moves as if it were stuck to the center point of a stick, the other end of the stick rotating around the joint/pivot point…

This constraint is of no use in most “standard” static or animated projects. However, you can use its results outside of the BGE, through the Game » Record Animation menu entry (from the main menu of the User Preferences window, see Rigid Bodies for more info on this topic).

For a demo file that shows some of the different types, see: BGE-Physics-RigidBodyJoints.blend.

Options


Rigid Body J: Rigid Body Joint
Target: Target_object
Pivot Type: Generic 6 DoF
Child Object: Child_object
Linked Collision ✓ Display Pivot ✓
Pivot: X: 0.000 Y: 0.000 Z: 0.000
Axis: X: 0° Y: 0° Z: 0°
Limits:
X Min: 0.000 Max: 0.000 | Y Min: 0.000 Max: 0.000 | Z Min: 0.000 Max: 0.000
Angle X Min: 0° Max: 0° | Angle Y Min: 0° Max: 0° | Angle Z Min: 0° Max: 0°

Rigid Body Joint panel

Target
: This constraint uses one target, and is not functional (red state) when it has none.

Joint Type

- Ball
  : works like an ideal ball-and-socket joint, i.e. allows rotations around all axes like a shoulder joint.
- Hinge
  : works in one plane, like an elbow: the owner and target can only rotate around the X axis of the pivot (joint point).

  Limits
  : Angular limits for the X axis
- Cone Twist
  : similar to Ball, this is a point-to-point joint with limits added for the cone and twist axis

  Limits
  : Angular limits
- Generic 6DOF
  : works like the *Ball* option, but the target is no longer constrained at a fixed distance from the pivot point, by default (hence the six degrees of freedom: rotation and translation around/along the three axes).
  In fact, there is no longer a joint by default, with this option, but it enables additional settings which allow you to restrict some of these DoF:

  Limits
  : Linear and angular limits for a given axis (of the pivot) in Blender Units and degrees respectively.

Child Object
: normally, leave this blank. You can reset it to blank by right clicking and selecting Reset to Default Value.

Linked Collision
: When enabled, this will disable the collision detection between the owner and the target (in the physical engine of the BGE).

Display Pivot
: When enabled, this will draw the pivot of the joint in the 3D views. Most useful, especially with the Generic 6DOF joint type!

Pivot
: These three numeric fields allow you to relocate the pivot point, *in the owner's space*.

Axis
: These three numeric fields allow you to rotate the pivot point, *in the owner's space*.

Script Constraints

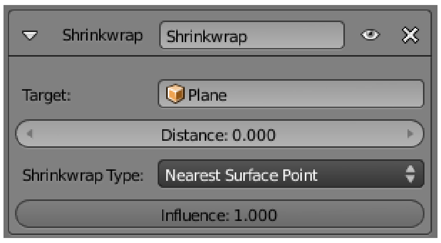
This feature is not supported in Blender 2.6

Shrinkwrap Constraint

The Shrinkwrap constraint is the “object counterpart” of the *Shrinkwrap* modifier. It moves the owner origin and therefore the owner object's location to the surface of its target.

This implies that the target *must* have a surface. In fact, the constraint is even more selective, as it can only use meshes as targets. Hence, the *Shrinkwrap* option is only shown in the *Add Constraint to Active Object* menu, CtrlAltC, (or its bone’s equivalent), when the selected inactive object is a mesh.

Options



Shrinkwrap panel

Target
: This constraint uses one target, which *must be a mesh object*, and is not functional (red state) when it has none.

Distance
: This numeric field controls the offset of the owner from the shrunk computed position on the target’s surface.
Positive values place the owner “outside” of the target, and negative ones, “inside” the target.
This offset is applied along the straight line defined by the original (i.e. before constraint) position of the owner, and the computed one on the target’s surface.

Shrinkwrap Type
: This drop-down list allows you to select which method to use to compute the point on the target’s surface to which to translate the owner’s center. You have three options:

  Nearest Surface Point
  : The chosen target’s surface’s point will be the nearest one to the original owner’s location. This is the default and most commonly useful option.

  Projection
  : The target’s surface’s point is determined by projecting the owner’s center along a given axis.
  This axis is controlled by the three X, Y and Z toggle buttons that show up when you select this type. This mean the projection axis can only be aligned with one of the global axes, median to both of them (XY, XZ or YZ), or to the three ones (XYZ).
  When the projection of the owner’s center along the selected direction does not hit the target’s surface, the owner’s location is left unchanged.

  Nearest Vertex
  : This method is very similar to the *Nearest Surface Point* one, except that the owner’s possible shrink locations are limited to the target’s vertices.

Animation

Introduction

Animation is making an object move or change shape over time. Objects can be animated in many ways:

Moving as a whole object
: Changing their position, orientation or size in time;

Deforming them
: animating their vertices or control points;

Character Animation via Armature
: animated to deform by the movement of bones inside the mesh, a very complex and flexible interaction that makes character-shaped objects appear to walk and jump.

In this chapter we will cover the first two, but the basics given here are actually vital for understanding the following chapters as well.

Three methods are normally used in animation software to make a 3D object move:

Key frames
: Complete positions are saved for units of time (frames). An animation is created by interpolating an object fluidly through the frames. The advantage of this method is that it allows you to work with clearly visualized units. The animator can work from one position to the next and can change previously created positions, or move them in time.

Animation Curves
: Curves are interpolated from keyframes, and can be drawn for each XYZ component for location, rotation, and size, as well as any other attribute in Blender. These form the graphs for the movement, with time set out horizontally and the value set out vertically. The advantage of this method is that it gives you precise control over the results of the movement.

Path
: A curve is drawn in 3D space, and the Object is constrained to follow it according to a given time function of the position along the path.

The first two systems in Blender are completely integrated in a single one, the F-Curve system.

In Blender 2.5x, everything can now be animated. Previously, only certain datablock had the ability to be keyframed. Now users have the ability to animate nearly any type of data that can be changed to multiple values.

Animation features and tools overview

The BSoD Introduction to Character Animation tutorial is a good starting point for learning character animation. Even if you never used Blender before.

Animation Basics

Actions
: Actions are used to record the animation of objects and properties.

Drivers
: Drivers are used to control and animate properties.

Keying Sets
: Keying Sets are used to record a set of properties at the same time.

Markers
: Markers are used to mark key points/events within an animation.

Motion Paths
: Motion Paths are used to visualize an animation.

Shape Keys
: Shape Keys are used to deform objects into new shapes.

Animation Editors

Timeline
: The Timeline Editor is a quick editor to set and control the time frame.
: This also has some tools for animation.

Graph Editor
: The Graph Editor is mostly used to edit the F-Curves and Keyframes for Channels and Drivers.
: F-Curves
: F-Curve Modifiers

Dope Sheet
: The Dopes Sheet contains a collection of animation editors.

NLA Editor
: The NLA Editor is used to edit and blend Actions together.

Categories

Modifiers
: Modifiers are automatic operations that affect an object in a non-destructive way.
: With modifiers, you can perform many effects automatically that would otherwise be tedious to do manually.

Rigging
: Rigging.

Constraints

Constraints are a way of connecting transform properties (position, rotation and scale) between objects.

Physical Simulation
: This category covers various advanced Blender effects, often used to simulate real physical phenomena.
There is the Particle System for things like hair, grass, smoke, flocks.
Soft Bodies are useful for everything that tends to bend, deform, in reaction to forces like gravity or wind.
Cloth simulation, to simulate clothes or materials.
Rigid Bodies can simulate dynamic objects that are fairly rigid.
Fluids, which include liquids and gasses, can be simulated, including Smoke.
Force Fields can modify the behavior of simulations.

Motion Tracking
: Motion tracking is a new technique available in Blender. It is still under development, and currently supports basic operations for 2D motion tracking, 3D motion tracking, and camera solution.

Animation Scripts
: Addon scripts for animation.

Rigging Scipts
: Addon scripts for rigging.

Animation Techniques and Deformations

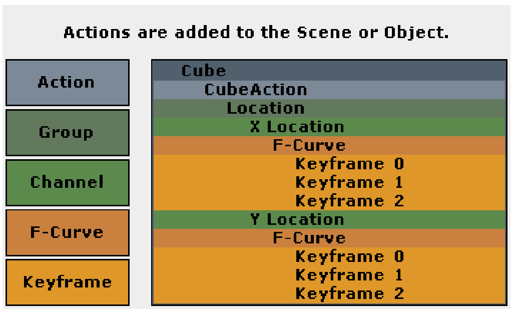
- Constraints
- Moving objects on a Path
- Game Engine Physics Recording
- Methods of deformation
- Shape Keys
- Deforming by a Lattice
- Deforming with Hooks

> See also Hooks - Uses a modifier as a way to change the shape of a mesh. Sorta like sticking a fish hook in a mesh and pulling. Uses the principles discussed in Shape Keys.

Actions

Actions

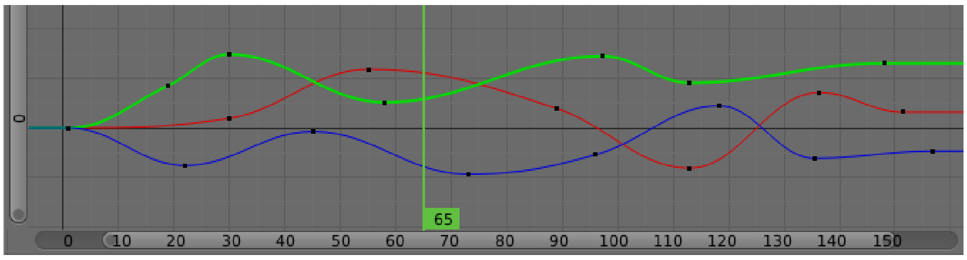
When animating objects and properties in blender, Actions record and contain the data.



Actions.

So when you animate an object by changing its location with keyframes, the animation is saved to the Action.

Each property has a channel which it is recorded to, for example, Cube.location.x is recorded to Channel X Location.



Graph Editor. Each Channel has an F-Curve represented by the lines between the keyframes.

Actions
: Record and contain animation data.

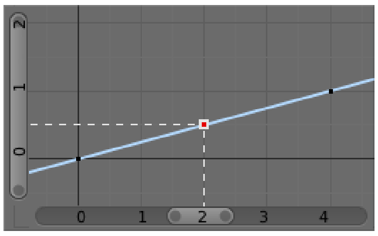
Groups
: Are groups of channels.

Channels
: Record properties.

F-Curves
: Are used to interpolate the difference between the keyframes.

Keyframes
: Are used to set the values of properties.

F-Curve Interpolation



Graph Editor: Channel F-Curve.

The keyframes are set values by the user.

The *F-Curve* is used to interpolate the difference between the keyframes.

The *F-Curve* has different types of interpolation and also *F-Curve Modifiers*.

Most the settings for the *F-Curve* are found in the *Graph Editor*.

Basic Animation

These are some common ways to animate objects.
These methods can be used on different objects, like armature bones in pose mode.

Insert Keyframes

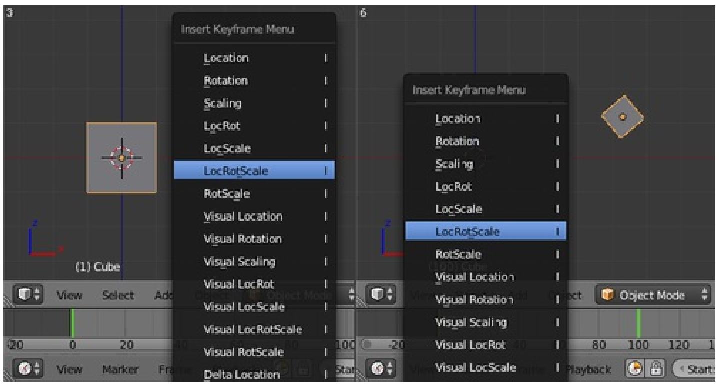
This example shows you how to animate a cubes location, rotation, and scale.

1. First, in the *Timeline*, or other animation editors, set the frame to 1.

2. With the *Cube* selected in *Object Mode*, press I in the 3D View.
3. From the *Insert Keyframe Menu* select *LocRotScale*.

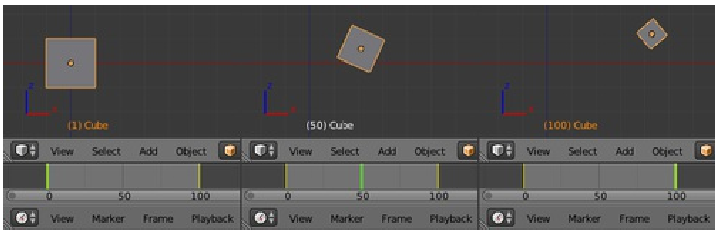
   This will record the location, rotation, and scale, for the *Cube* on frame 1.

4. Set the frame to 100.
5. Use Grab/Move G, Rotate R, Scale S, to transform the cube.
6. Press I in the 3D View. From the *Insert Keyframe Menu* select *LocRotScale*.



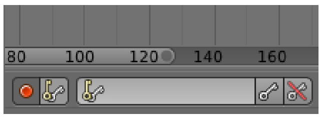
3. 6. Insert Keyframes.

To test the animation, press AltA to play.



The animation on frames 1, 50, 100.

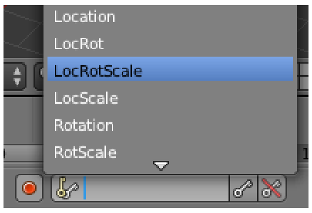
Auto Keyframe


Timeline Auto Keyframe.

Auto Keyframe is the red record button in the *Timeline* header. Auto Keyframe adds keyframes automatically to the set frame if the value for transform type properties changes.

See Timeline V Keyframe Control for more info.

Keying Sets



Timeline Keying Sets.

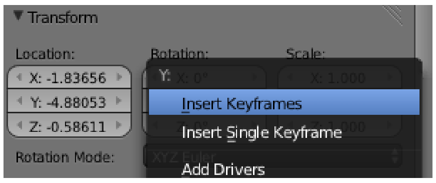
Keying Sets are a set of keyframe channels. They are used to record multiple properties at the same time. There are some built in keying sets, 'LocRotScale', and also custom keying sets can be made.

To use the keying set, first select a keying set from the *Timeline* header, or the *Keying Sets Panel*.

Now when you press I in the 3D view, blender will add keyframes for all the properties in the active keying set.

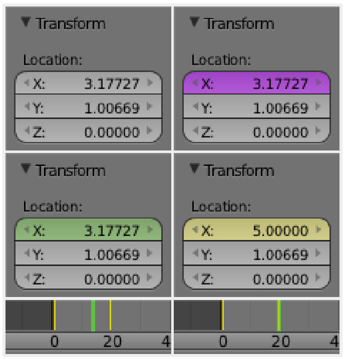
See Keying Sets for more info.

Properties



Keyframe properties.

Keyframes can be used to animate lots of different properties in blender.
To add keyframes to a property in the UI, RMB the property, then select Insert Single Keyframe, or Insert Keyframes.
Insert Keyframes I will add a keyframes for the set of properties.



Properties, Drivers, Keyframes.

Properties have different colors and menu items for different states.

Gray - Property is not animated with Keyframes or Drivers.
    Insert Keyframes I.
    Insert Single Keyframe.
    Add Drivers.
    Add Single Driver.
    Paste Driver.

Purple - Property value is controlled with a Driver.
    Delete Drivers.
    Delete Single Driver.
    Copy Driver.
    Paste Driver.

Green - Property has Channel with Keyframes.
    Insert Keyframes I.
    Insert Single Keyframe.
    Clear Keyframes Alt⇧ ShiftI
    Clear Single Keyframes.

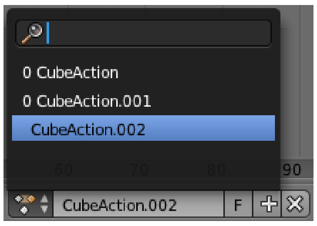
Yellow - Property has Keyframes on the current Frame.
    Replace Keyframes I.
    Replace Single Keyframe.
    Delete Keyframes AltI.
    Delete Single Keyframe.
    Clear Keyframes Alt⇧ ShiftI
    Clear Single Keyframes.

Each property also has some Keying Set options.
    Add All to Keying Set K.
    Add Single to Keying Set.
    Remove from Keying Set.

Editing

3D View.
    Insert Keyframes on current frame I
    Delete Keyframes on current frame AltI

Working with Actions



Action Browser.

When you first animate an object by adding keyframes, blender creates an *Action* to record the data.

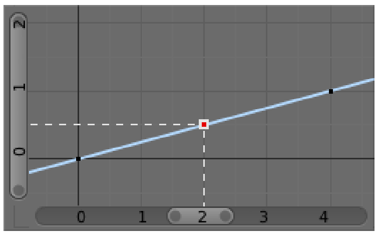
Actions can be managed with the *Action Browser* in the *DopeSheet* *Action Editor* header, or the properties region of the *NLA Editor*.

If you are making multiple actions for the same object, press the **F** button for each action, this will give the actions a *Fake User* and will make blender save the unlinked actions.

Objects can only use one *Action* at a time for editing, the *NLA Editor* is used to blend mutiple actions together.

Drivers

Drivers


Graph Editor: Driver example.

Drivers can use properties, numbers, transformations, and scripts, to control the values of properties.

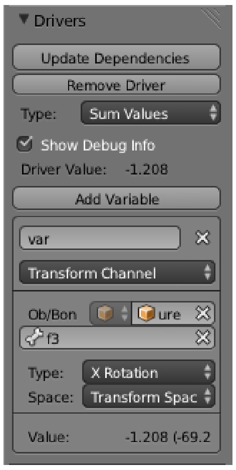
Using a F-Curve, the driver reads the value of the Driver Value and sets the value of the selected property it was added to.

So from this example, if the Driver Value is 2.0 the property will be 0.5.

The Driver Value is determined by Driver Variables or a Scripted Expression.

Most the settings for the drivers F-Curves are found in the Graph Editor.

Drivers Panel



Graph Editor: Drivers:
Drivers Panel.

This panel is located in the Graph Editor with the mode set to Drivers.

The drivers panel is for setting up *Driver Variables* or a *Scripted Expression* which will determine the value of the *Driver Value*.

Driver Settings

Update Dependencies
: This will force an update for the Driver Value dependencies.

Remove Driver
: Removes the driver from the object.

Type
: The type of calculation to use on the set of Driver Variables. (If you only have one driver variable there is no real difference between average, sum, minimum and maximum)

  Average Value
  : Uses the the average value of the referenced Driver Variables.

  Sum Values
  : Uses the the sum of the referenced Driver Variables.

  Scripted Expression
  : Uses a Scripted Expression. See Expr.

    You must write a python expression which performs your own calculations on the Driver Variables.

  Minimum Value
  : Uses the lowest value from the referenced Driver Variables.

  Maximum Value

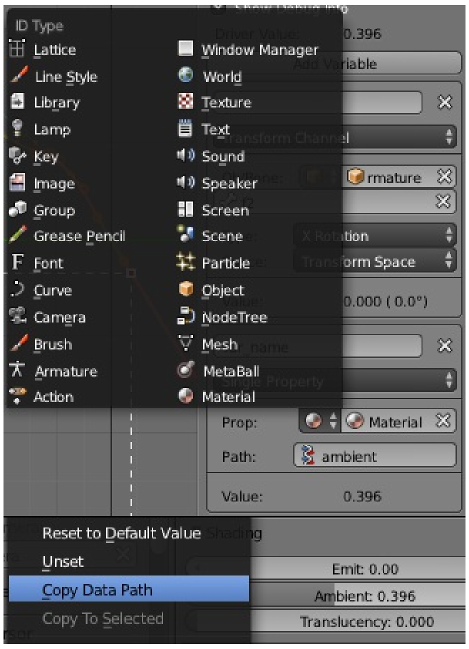
Uses the highest value from the referenced Driver Variables.

Expr
: Scripted Expression.
Here you can add real numbers, math operators, math functions, python properties, driver functions.
See Driver Expression below for some examples.

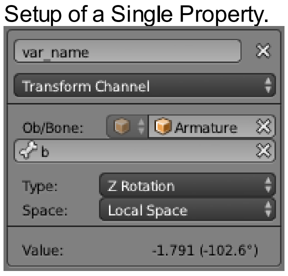
Show Debug Info
: Shows the Driver Value.

The current value of the variables or scripted expression.

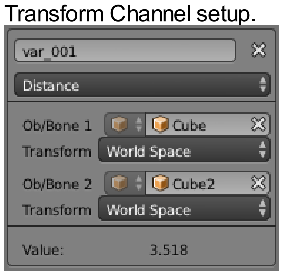
Add Variable
: Adds a new Driver Variable.



Setup of a Single Property.



Transform Channel setup.



Distance setup.

Driver Variables

Name
: Name to use for scripted expressions/functions.

No spaces or dots are allowed and must start with a letter.

Variable Type
: The type of variable to use.

Single Property
: Use the value from some RNA property.

For example, the Ambient shading color from a material.
First select the type of ID-block, then the ID of the ID-block, then copy and paste an RNA property (Ctrl+V).

ID-Type
: The ID-Block type, example, Key, Image, Object, Material.

ID
: The ID of the ID-Block type, example, Material.001.

RNA Path
: The RNA id name of the property, example, 'ambient' from material shading.

Transform Channel
: Use one of the Transform channels from an object or bone.

ID
: ID of the object, example, Cube, Armature, Camera.

Bone
: ID of the Armature bone, example, Bone, Bone.002, Arma.r.

This option is for armatures.

Type
: Example, X Location, X Rotation, X Scale.

Space
: World Space, Transform Space, Local Space.

Rotational Difference
: Use the rotational difference between two objects or bones.

Distance
: Use the distance between two objects or bones.

Value
: Shows the value of the variable.

Workflow

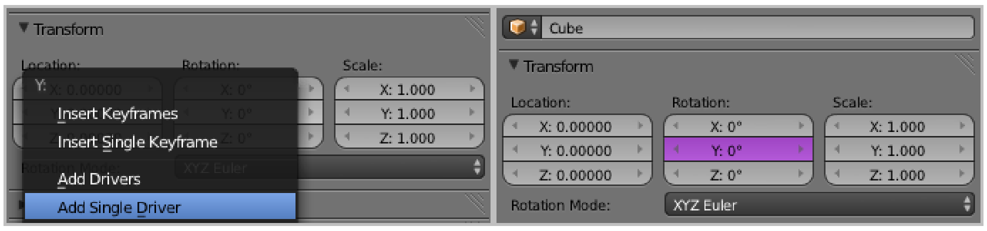
Adding Drivers

To control a property with a driver, find the property you want to add driver to.

RMB the property and select one of the following options.

Add Drivers
: This will add drivers to the set of properties related to the selected one.
For example, it will add drivers to X, Y, and Z for Rotation.

Add Single Driver
: This will add a single driver to the selected property.



Add Single Driver.

Transform Driver

This example shows you how rotate a cube mesh by moving another cube left or right in the 3D view.
First make sure you are in the *Front Orthographic View*Num1, Num5.

1. In *Object Mode* select then *Duplicate* ⇧ ShiftD the default Cube.

   Move the cube to a new location. You should have two mesh objects, *Cube* and *Cube.001*.

2. With *Cube.001* selected as the active object, *Add Single Driver* to the *Rotation Y* property.
3. Open the *Graph Editor*, set the mode to *Drivers*.

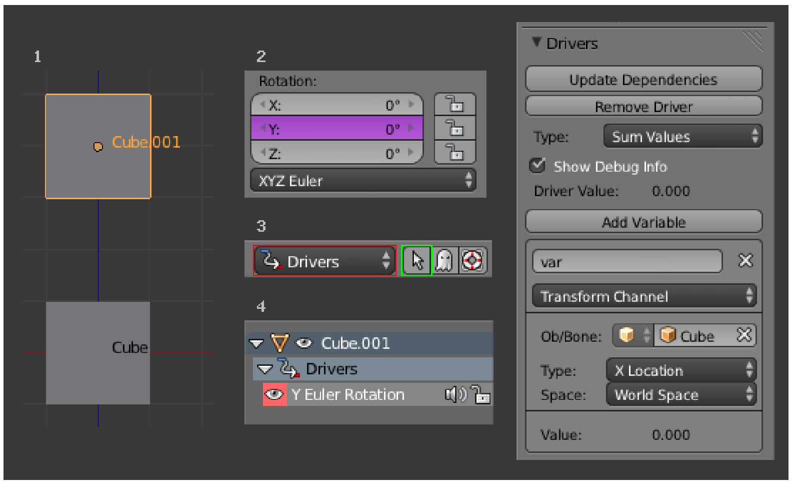
   Show Only Selected is useful disabled for drivers, marked in green.

4. Open the Properties Region N, go to the *Drivers Panel*.

   You may need to select the driver *Y Euler Rotation* LMB for the *Drivers Panel* to appear.

5. Set the driver *Type* to *Sum Values*.
6. Set the driver variable *var* settings.

Set *Type* to *Transform Channel*.
Set *Ob/Bone ID-block* to *Cube*.
Set *Transform Type* to *X Location*.
Set *Transform Space* to *World Space*.



Transform Driver workflow.

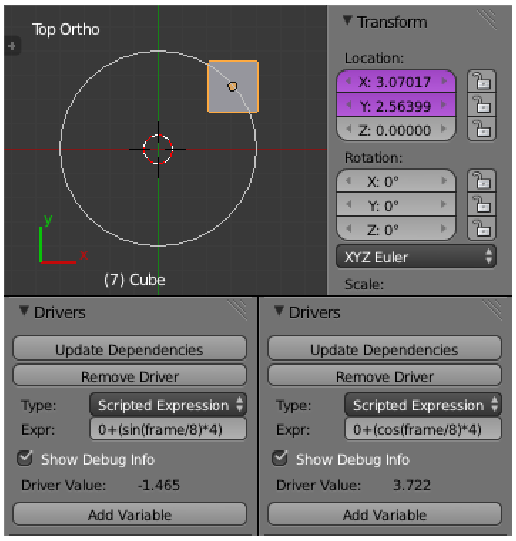
Now when you move the *Cube* left or right in the 3D View, *Cube.001* should rotate.

Examples

Some Driver Examples.

Driver Expression

Here are some examples using the scripted expression Expr to set the Driver Value.



Object Rotation.

Orbit a point

Here two drivers have been added to the Cube, X Location and Y Location.

The scripted expressions are being used to set the object location.

X Location Expr
: **0+(sin(frame/8)*4)**

  (frame/8): is the current frame of the animation, divided by 8 to slow the orbit down.
  (sin( )*4): This returns the sine of (frame/8), then multiplies by 4 for a bigger circle.
  0+: is used to control the X Location offset of the orbit.

Y Location Expr
: **0+(cos(frame/8)*4)**

  (frame/8): is the current frame of the animation, divided by 8 to slow the orbit down.
  (cos( )*4): This returns the cosine of (frame/8), then multiplies by 4 for a bigger circle.
  0+: is used to control the Y Location offset of the orbit.

frame is the same as bpy.context.scene.frame_current.

Driver Namespace

There is a list of built in driver functions and properties.
These can be displayed via the python console.

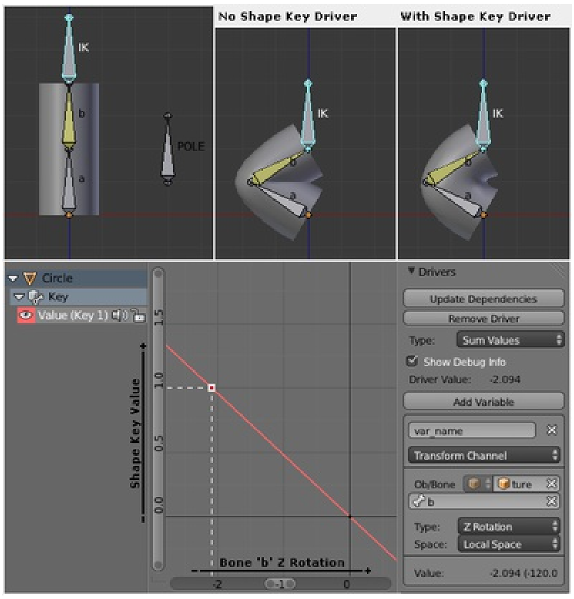
```
>>> bpy.app.driver_namespace['
                              __builtins__']
                              __doc__']
                              __loader__']
                              __name__']
                              __package__']
                              acos']
                              acosh']
                              asin']
                              asinh']
                              atan']
                              atan2']
                              atanh']
                              bpy']
                              ceil']
                              copysign']
                              cos']
                              cosh']
                              ..
```

This script will add a function to the driver namespace, which can then be used in the expression **driverFunc(frame)**.

```
import bpy

def driverFunc(val):

    return val * val    # return val squared

bpy.app.driver_namespace['driverFunc'] = driverFunc    # add function to driver_namespace
```

Shape Key Driver

This example is a Shape Key Driver. The driver was added to the shape key Value.



Type: Sum Values
Show Debug Info
Driver Value: -2.094
var_name
Transform Channel
Ob/Bone ture
b
Type: Z Rotation
Space: Local Space
Value: -2.094 (-120.0

This example uses the Armature Bone 'b' Z Rotation to control the Value of a Shape Key.
The bone rotation mode is set to XYZ Euler.

The Driver F-Curve is mapped like so
- Bone Z Rotation 0.0(0.0): Shape Key value 0.0
- Bone Z Rotation -2.09(-120.0): Shape Key value 1.0

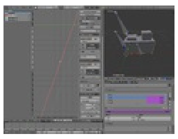
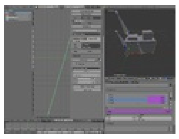
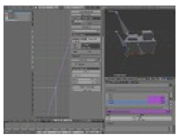
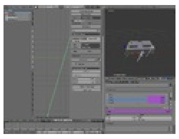
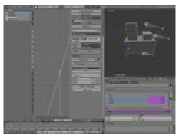
This kind of driver can also be setup with the Variable Type Rotational Difference.

See Shape Keys for more info.

Drivers And Multiple Relative Shape Keys

The following screenshots illustrate combining shape keys, bones, and drivers to make multiple chained relative shape keys sharing a single root. While it lacks the convenience of the single Evaluation Time of an absolute shape key, it allows you to have more complex relationships between your shape keys.

Drivers and Multiple Shape Keys

 Key1 must handle conflicting values from the two bones	 Key2A has different generator coefficients so it is activated in a different range of the bone's position.	 Key2B is the same as Key2A, but is controlled by the second bone.	 |when both bones are low, Key2B and Key2A are deactivated and Key1 is at low influence.	

The Basis shape key has the stacks fully retracted. Key1 has the base fully extended. Key2A has the left stack fully extended. Key2B has the right stack fully extended. Key2A and Key2B are both relative to Key1 (as you can see in the field in the bottom right of the Shape Keys panel.

The value of Key1 is bound to the position of bones by a driver with two variables. Each variable uses the world Z coordinate of a bone and uses the maximum value to determine how much the base should be extended. The generator polynomial is crafted such that the top of the dominant stack should line up with the bone for that stack.

The value of Key2A is bound to the position of bone.L . Its generator parameters are crafted such that when Key1's value reaches 1, the value of Key2A starts increasing beyond zero. In this way the top of the left stack will move with bone.L (mostly).

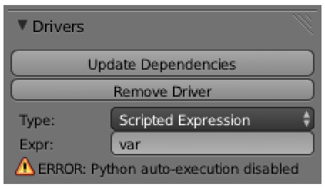
The value of Key2B is bound to the position of bone.R . Its generator parameters are similar to Key2A so that the top of the right stack will move with bone.R (mostly).

Since it's quite easy for bone.L and bone.R to be in positions that indicate conflicting values for Key1 there will be times when the bones do not line up with the tops of their respective stacks. If the driver for Key1 were to use Average or Minimum instead of Maximum to determine the value of the shape key then "conflicts" between bone.L and bone.R would be resolved differently. You will chose according to the needs of your animation.

Troubleshooting

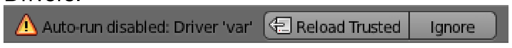
Some common problems people may run in to when using drivers.

Scripted Expression



Type: Scripted Expression
Expr: var
ERROR: Python auto-execution disabled

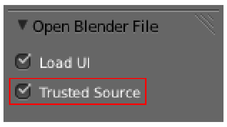
Graph Editor > Properties > Drivers.


Auto-run disabled: Driver 'var'

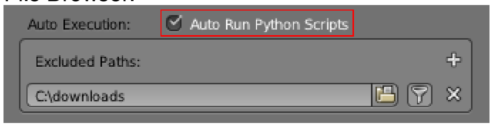
Info Header.

By default blender will not auto run python scripts.

If using a *Scripted Expression* Driver Type, you will have to open the file as *Trusted Source*, or set *Auto Run Python Scripts* in *User Preferences > File > Auto Execution*.



File Browser.


Auto Execution: Auto Run Python Scripts
Excluded Paths:
C:\downloads

User Preference > File > Auto Execution.

Rotational Properties are Radians

Parts of the User Interface may use different units of measurements for angles, rotation. In the Graph Editor while working with Drivers, all angles are Radians.

Intra-armature Bone Drivers Can Misbehave

There is a well known limitation with drivers on bones that refer to another bone in the same armature. Their values can be incorrectly calculated based on the position of the other bone as it was *before* you adjust the current_frame. This can lead to obvious shape glitches when the rendering of frames has a jump in the frame number (either because the .blend file is currently on a different frame number or because you're skipping already-rendered frames).

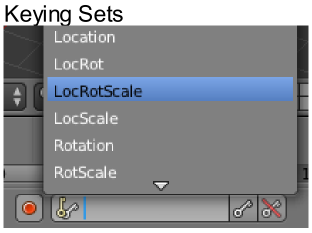
See Also

- Animation
- Graph Editor
- F-Curves
- Extending Blender with python.

Links

- Python and its documentation.
- functions.wolfram.com

Keying Sets



Timeline Keying Sets.

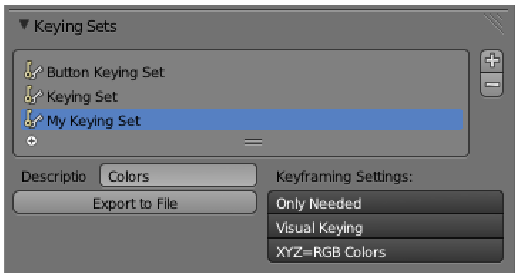
Keying Sets are a collection of properties. They are used to keyframe multiple properties at the same time, usually by pressing I in the 3D View.

There are some built in Keying Sets, and also custom Keying Sets called *Absolute Keying Sets*.

To select and use a Keying Set, set the *Active Keying Set* in the Timeline Header, or the *Keying Set Panel*, or press CtrlAlt⇧ ShiftI in the 3D View.

Keying Set Panel

This panel is used to add, select, manage *Absolute Keying Sets*.



Properties > Scene > Keying Set Panel.

Keying Set Name
: The active Keying Set is highlighted in blue, LMB X2 to rename.

+
: Add new (Empty) keying set to the active Scene.

-
: Remove the active Keying Set.

Active Keying Set properties

Description
: A short description of the keying set.

Export to File
: Export Keying Set to a python script *File.py*.
To re add the keying set from the *File.py*, open then run the *File.py* from the Text Editor.

Keyframing Settings

These options control all properties in the Keying Set.
Note, the same settings in *User Preferences* override these settings if enabled.

Only Needed

Only insert keyframes where they're needed in the relevant F-Curves.

Visual Keying

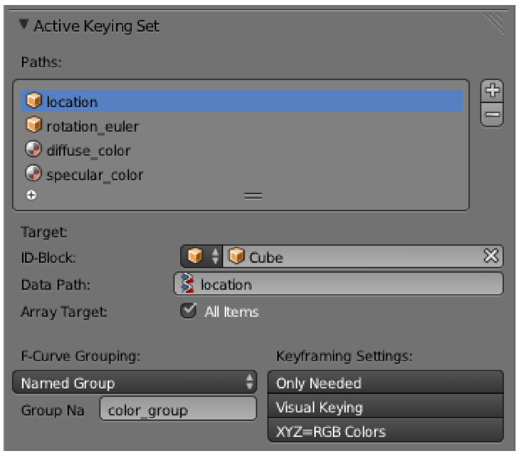
Insert keyframes based on the visual transformation.

XYZ=RGB Colors

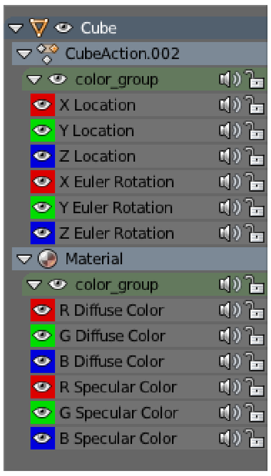
For new F-Curves, set the colors to RGB for the property set, Location XYZ for example.

Active Keying Set Panel

This panel is used to add properties to the active Keying Set.



Properties > Scene > Active Keying Set Panel.



Properties > Graph Editor > Channels, Named Group.

Paths
: A collection of *Paths* each with a *Data Path* to a property to add to the active Keying Set. The active *Path* is highlighted in blue.

+
: Add new empty path to active Keying Set.

-
: Remove active path from the active Keying Set.

Active Path properties

ID-Block
: Set the *ID-Type* + *Object ID Data Path* for the property.

Data Path
: Set the rest of the *Data Path* for the property.

Array Target
: Use *All Items* from the *Data Path* or select the array index for a specific property.

F-Curve Grouping
: This controls what *Group* to add the *Channels* to.
Keying Set Name, *None*, *Named Group*.

Keyframing Settings

These options control individual properties in the Keying Set.

Only Needed

Only insert keyframes where they're needed in the relevant F-Curves.

Visual Keying

Insert keyframes based on the visual transformation.

XYZ=RGB Colors

For new F-Curves, set the colors to RGB for the property set, Location XYZ for example.

Adding Properties

Some ways to add properties to keying sets.

RMB the property in the *User Interface*, then select *Add Single to Keying Set* or *Add All to Keying Set*. This will add the properties to the active keying set, or to a new keying set if none exist.

Hover the mouse over the properties, then press K, to add *Add All to Keying Set*.

See Also

- Timeline Header - V Keyframe Control

Markers

Markers are used to denote frames at which something significant happens – it could be that a character's animation starts, the camera changes position, or a door opens, for example. Markers can be given names to make them more meaningful at a quick glance. They are available in many of Blender's windows, under different forms. Unlike the keyframes, markers are always placed at a whole frame number, you cannot e.g. set a marker at "frame **2.5**".

Markers can be created and edited in all of the following editors (including their different modes):

- The Timeline.
- The Graph Editor.
- The The Dope Sheet.
- The NLA Editor.
- The Video Sequence Editor.

A marker created in one of these windows will also appear in all others that support them, including:

- The 3D View window.

There are Timeline Markers and Pose Markers in Blender.

Timeline Markers

Timeline markers are used to:

- Switching multiple cameras.
- Synchronization timing keyframes with markers.

Pose markers

Pose markers is another type of markers, that are specific to the armatures and are used to denote poses in the *Action Editor* mode of Dope Sheet editor.

Click the Action Editor menu Marker > Show Pose Markers to see pose markers.

With this enabled, creating markers will add only *pose* markers in the Action Editor!

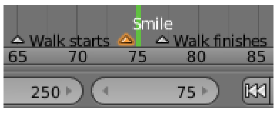
With Show Pose Markers disabled, you may select and edit only *timeline* markers in the Action Editor.

Pose markers are related to the pose libraries, and are discussed in detail here.

Timeline markers is possible to convert to the Pose markers with Marker > Make Markers Local. Note that the original Timeline marker *will be gone*. If you want to keep it, make a duplicate before you convert.

Visualization

Standard

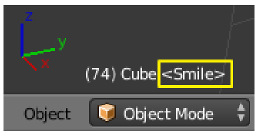


Timeline markers: small but useful.

Most of the window types visualize markers the same way: as small triangles at their bottom, white if unselected or yellow if selected.

If they have a name, this is shown to their right, in white when the marker is selected. See (*Timeline markers: small but useful*).

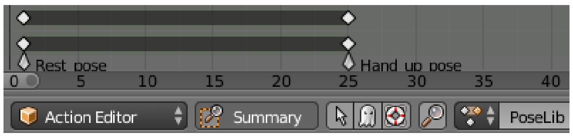
3D View



Marker in a 3D View.

The *3D View*windows do not allow you to create/edit/remove markers, they just show their name between <> at there bottom left corner, near the active object's name, when you are at their frame (see *Marker in a 3D view*).

Pose Markers



Pose markers in the Action Editor mode of the Dope

Sheet editor.

Pose markers show a diamond-shaped icon in *Action Editor* mode of Dope Sheet editor. See (*Pose markers in the Action Editor mode of the Dope Sheet editor*).

Creating and Editing Markers

Pose markers are created automatically at the moment adding new pose in Pose Library. Methods of the editing are same for both Timeline and Pose markers.

Creating Timeline Markers

Mode: all modes

Hotkey: M

Menu: Marker » Add Marker

The simplest way to add a marker is to move to the frame where you would like it to appear, and press M.

Alternatively, you can press AltA (or the "playback" button of the *Timeline* window) to make the animation play, and then hit M at the appropriate points. This can be especially useful to mark the beats in some music.

Selecting Markers

Mode: all modes

Hotkey: RMB, ⇧ Shift RMB, A, B

Click RMB on the marker's triangle to select it. Use ⇧ Shift RMB to (de)select multiple markers.

To (de)select all markers use A. There are using border (de)select them with B.

The corresponding options are found in the *Select* menus of the editors, where the markes are using.

Naming Markers

Mode: all modes

Hotkey: CtrlM

Menu: Marker » Rename Marker

Having dozens of markers scattered throughout your scene's time won't help you much unless you know what they stand for. You can name a marker by selecting it, pressing CtrlM, typing the name, and pressing the OK button.

Moving Markers

Mode: all modes

Hotkey: G

Menu: Marker » Grab/Move Marker

Once you have one or more markers selected, hit G to move them, and confirm the move with LMB or ↵ Enter (as usual, cancel the move with RMB, or Esc).

By default, you grab the markers in one-frame steps, but if you hold Ctrl, the markers will move in steps corresponding (defined by Frame Rate option in Dimensions panel of Render context) to one second – so if you have set your scene to **25 fps**, the markers will move in twenty-five-frames steps.

Duplicating Markers

Mode: all modes

Hotkey: ⇧ ShiftD

Menu: Marker » Duplicate Marker

You can duplicate the selected markers by hitting ⇧ ShiftD. Once duplicated, the new ones are automatically placed in grab mode, so you can move them where (or rather when) you want.

Deleting Markers

Mode: all modes

Hotkey: X

Menu: Marker » Delete Marker

To delete the selected marker(s) simply press X and confirm the pop-up message with LMB.

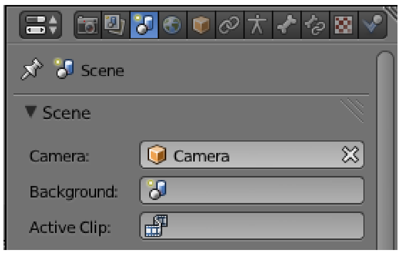
Using Timeline Markers for switching Multiple Cameras

To change cameras mid-animation, you need to use markers.

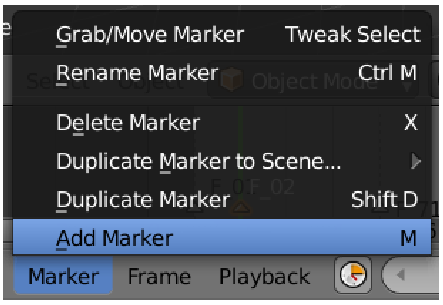
Note that markers behave like keyframes, so you will need *at least two* markers with *two cameras* bound to them to have any camera switching.

For quickly "bind camera to marker" by following the step-by-step below:

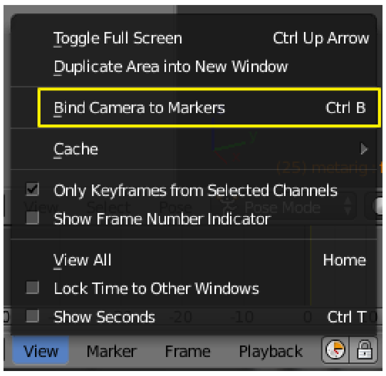
- Select your camera (to be binded to timeline) in the 3D view with RMB-click or with Object Data context for camera in Properties window.


Scene
Scene
Camera: Camera
Background:
Active Clip:

- Hover over the timeline (make sure mouse is within Timeline panel), jump to a specific frame, create a marker there by hitting M on the keyboard or select corresponding item - Marker » Add Marker.



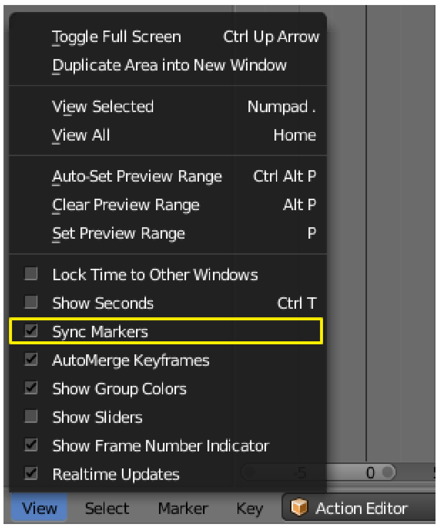
- Now, while still hovering on the timeline, press CtrlB to Bind Camera to Markers or select corresponding item - View » Bind Camera to Markers.


Toggle Full Screen Ctrl Up Arrow
Duplicate Area into New Window
Bind Camera to Markers Ctrl B
Cache
Only Keyframes from Selected Channels
Show Frame Number Indicator
View All Home
Lock Time to Other Windows
Show Seconds Ctrl T
View
Marker
Frame
Playback

- Continue with next camera.
- Now, if you jump to inside camera (by hitting numpad Zero), it will switch camera specified by marker as you scrub along the timeline.

Switching Cameras.

Synchronization timing keyframes with markers



Option Sync Markers in View menu of the Action Editor mode of the Dope Sheet editor.

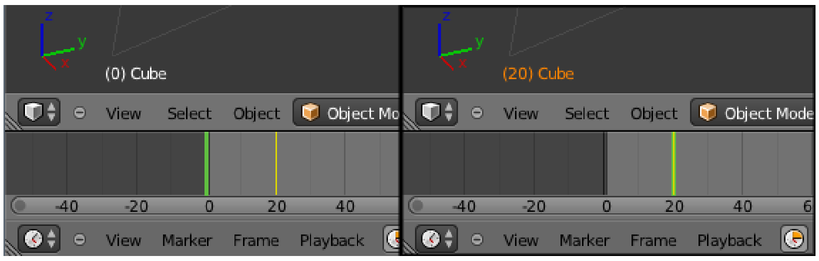
Anothe the usage Timeline markers is synchronization timing keyframes with markers in the Dope Sheet and Video Sequence Editor (the option Sync Markers in View menu). This allows to tape editing or final cut yours actions, sequences, scenes, movie and audio clips.

Keyframe Visualization

There are some important visualization features in the 3D views that can help animation.

Keyframe Visualization

When the current frame is a keyframe for the current active object, the name of this object (shown in the bottom left corner of the 3D views) turns yellow.

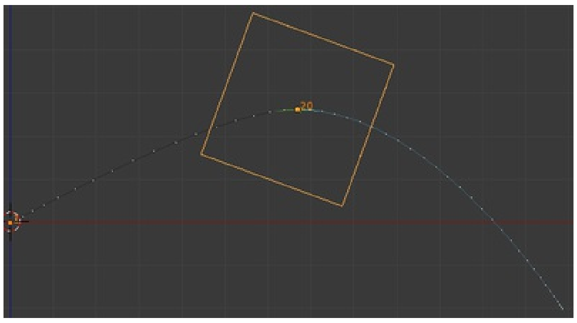

(0) Cube
View Select Object Object Mo
-40 -20 0 20 40
View Marker Frame Playback
(20) Cube
View Select Object Object Mode
-40 -20 0 20 40 6
View Marker Frame Playback
Left: Current frame at 0. Right: Current frame is a keyframe for Cube

Motion Paths

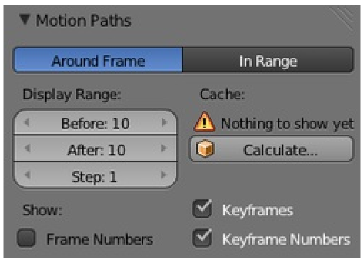
Mode: Object mode

Panel: Object

This feature allows you to visualize the animation of objects by displaying their position over a series of frames.



An animated cube with its motion path displayed


Motion Paths

Around Frame | In Range

Display Range: Before: 10, After: 10, Step: 1

Cache: Nothing to show yet, Calculate...

Show: Frame Numbers (unchecked), Keyframes (checked), Keyframe Numbers (checked)

Motion paths panel

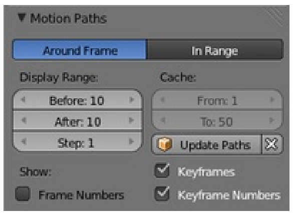
Before we look at its options (all regrouped in the same Visualisations panel, in the Editing context, let’s first see how to display/hide these paths. You have to do it manually – and you have to first select the objects you want to show/hide the motion paths. Then,

- To show the paths (or update them, if needed), click on the Calculate Path button.
- To hide the paths, click on the Clear Paths button

Remember: only selected object and their paths are affected by these actions!

The paths are drawn in black with white dots indicating frames, and a blue glow around the current frame.

Options


Motion Paths

Around Frame | In Range

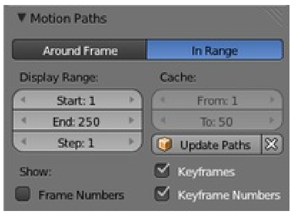
Display Range: Before: 10, After: 10, Step: 1

Cache: From: 1, To: 50, Update Paths

Show: Frame Numbers (unchecked), Keyframes (checked), Keyframe Numbers (checked)

The Motion Paths Panel set to "Around Frame"

Around Frame
: Around Frame, Display Paths of poses within a fixed number of frames around the current frame. When you enable this button, you rather get paths for a given number of frames before and after the current one (again, as with ghosts).



The Motion Paths Panel set to "In Range"

In Range
: In Range, Display Paths of poses within specified range.

Display Range
: Before/After

    Number of frames to show before and after the current frame (only for ‘Around Current Frame’ Onion-skinning method)

: Start/End

    Starting and Ending frame of range of paths to display/calculate (not for ‘Around Current Frame’ Onion-skinning method)

: Step

    This is the same thing as the GStep for ghosts – it allows you the only materialize on the path one frame each *n* ones. Mostly useful when you enable the frame number display (see below), to avoid cluttering the 3D views.

Frame Numbers
: When enabled, a small number appears next to each frame dot on the path, which is of course the number of the corresponding frame…

Keyframes
: When enabled, big yellow square dots are drawn on motion paths, materializing the keyframes of their bones (i.e. only the paths of keyed bones at a given frame get a yellow dot at this frame).

Keyframe Numbers
: When enabled, you’ll see the numbers of the displayed keyframes – so this option is obviously only valid when Show Keys is enabled.

Cache
: From/To

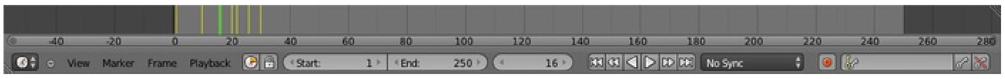
    These are the start/end frames of the range in which motion paths are drawn. You cannot modify this range without deleting the motion path first.

: Calculate Paths/ Update Paths

    If no paths have been calculated, Calculate Paths will create a new motion path in cache. In the pop up box, select the frame range to calculate.
    If a path has already been calculated, Update Paths will update the path shape to the current animation. To change the frame range of the calculated path, you need to delete the path and calculate it again.

The Timeline

The *Timeline* window, identified by a clock icon, is shown by default at the bottom of the screen.



The *Timeline* is not much of an editor, but more of a information and control window.

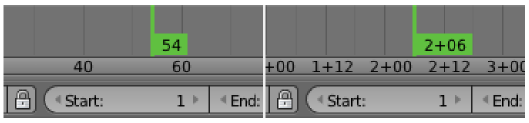
Here you can have an overview of the animation part of your scene
What is the current time frame, either in frames or in seconds, where are the keyframes of the active object, the start and end frames of your animation, markers, etc...

The *Timeline* has *Player Controls*, to play, pause the animation, and to skip though parts of the scene.

It also has some tools for *Keyframes*, *Keying Sets*, and *Markers*.

Timeline Elements

Time Cursor



Time Cursor

The *Time Cursor* is the green line, its used to set and display the current time frame.

The *Time Cursor* can be set or moved to a new position by pressing or holding LMB in the Timeline window.

The current frame or second can be displayed on the *Time Cursor*, check the View menu for settings.

The *Time Cursor* can be moved in steps by pressing Arrow left or Arrow right, or in steps of 10 frames by pressing ⇧ ShiftArrow up or ⇧ ShiftArrow down.

Keyframes

For the active and selected objects, keyframes are displayed as a yellow line.

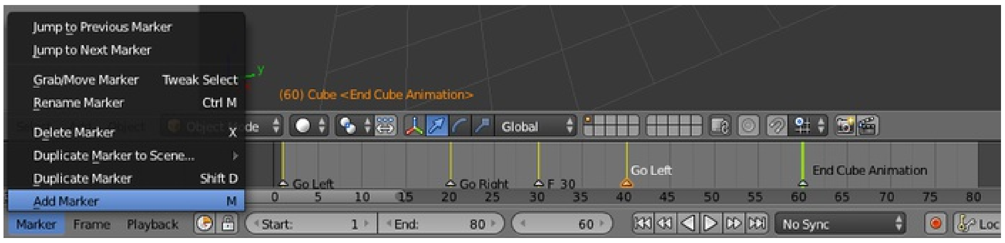
For *Armatures*, the object keyframes and the pose bones keyframes are drawn.

Only Selected Channels can be enabled. *Timeline > View> Only Selected Channels*.
For *Armatures*, this will draw the object keyframes, and the keyframes for the active and selected pose bones.

Markers

Markers are the small triangles, with their name near them.

Markers are usually used to identify key parts of the animation.



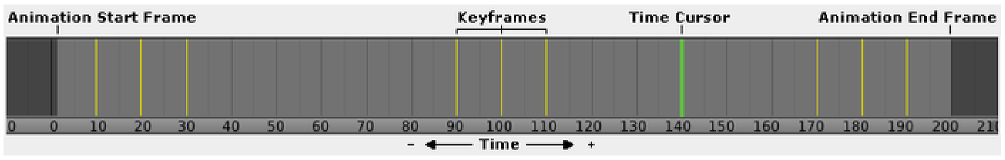
Markers can be selected by pressing RMB or ⇧ Shift RMB to select more.

See Marker Menu below or Markers for more info.

Adjusting the View

Timeline Area

The main *Timeline* area displays the animation frames over time.



The *Timeline* can be panned by holding MMB, then dragging the area left or right.

You can zoom the *Timeline* by using Ctrl MMB, the mouse Wheel, or pressing the - and + keys on the numpad.

By default, the *Playback/Rendering Range* (Frame Start 1 to Frame End 200) is a lighter shade of gray.
The start and end frame can be set to the *Time Cursor* by pressing S or E.
The *Playback Range* can also be set by pressing P then drawing a box.

Timeline Header

View Menu

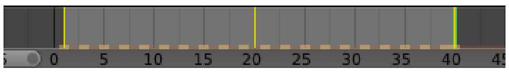
The *View Menu* controls what you see, and what it looks like.

Toggle Full Screen
Maximize or minimize the *Timeline* window. CtrlArrow up or CtrlArrow down

Duplicate Area into New Window
This creates a new OS window, and sets the editor window to the *Timeline*.

Bind Camera to Markers
This is used switch cameras during animation.
It binds the active camera to the selected markers.
First select a camera. Then select the marker(s). Then use the function. CtrlB

Cache
This will display the baked *Cache Steps* for the active object.



Timline Cache

Show Cache
Show all enabled types.

Softbody, *Particles*, *Cloth*, *Smoke*, *Dynamic Paint*, *Rigid Body*.

Only Selected Channels
For *Armatures*, this will draw the object keyframes, and the keyframes for the active and selected pose bones.

Show Frame Number Indicator
This will draw the current frame or seconds on the *Time Cursor*.

View All
Maximize the *Timeline* area based on the Animation Range. ↖ Home

Show Seconds
Show time in seconds for the *Timeline* and the the *Time Cursor* based on the FPS. CtrlT

Marker Menu

Jump to Previous Marker

Jump to Next Marker

Grab/Move Marker
Grab/Move the selected markers. G

Rename Marker
Rename the active marker. CtrlM

Delete Marker
Delete selected markers. X

Duplicate Marker to Scene...
Duplicate the selected markers to another scene.

Duplicate Marker
Duplicate the selected markers. ⇧ ShiftD

Add Marker
Add marker to the current frame. M

Frame Menu

Auto-Keyframing Mode
This controls how the Auto Keyframe mode works.
Only one mode can be used at a time.

Add & Replace
Add or Replace existing keyframes.

Replace
Only Replace existing keyframes.

Playback Menu

Audio Scrubbing

If your animation has sound, this option plays bits of the sound wave while you move the time cursor with LMB or keyboard <- arrows ->.

Audio Muted

Mute the sound from Sequence Editors.

AV-sync

Play back and sync with audio clock, dropping frames if frame display is too slow.
See Header Controls **IV** Synchronize Playback for more info.

Frame Dropping

Play back dropping frames if frames are too slow.
See Header Controls **IV** Synchronize Playback for more info.

Clip Editors

While playing, updates the *Movie Clip Editor*.

Node Editors

While playing, updates the Node properties for the *Node Editor*.

Sequencer Editors

While playing, updates the *Video Sequence Editor*.

Image Editors
Not sure what is updated, maybe gif images or, image sequence.

Image Editors

// Todo.

Property Editors

When the animation is playing, this will update the property values in the UI.

Animation Editors

While playing, updates the *Timeline*, *Dope Sheet*, *Graph Editor*, *Video Sequence Editor*.

All 3D View Editors

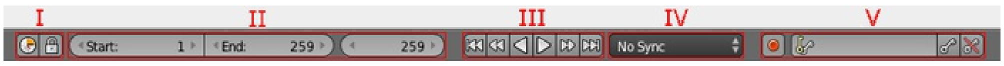
While playing, updates the *3D View* and the the *Timeline*.

Top-Left 3D Editor

While playing, updates the *Timeline* if *Animation Editors* and *All 3D View Editors* disabled.

Header Controls

The Timeline header controls.



Timeline header controls.

I Range Control

Use Preview Range
This is an alternative range used to preview animations.
This works for the UI playback, this will not work for rendering an animation.

Lock Time Cursor to Playback Range
This limits the *Time Cursor* to the *Playback Range*.

II Frame Control

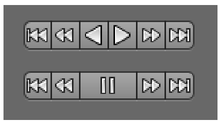
Start Frame
: The start frame of the animation / playback range.

End Frame
: The end frame of the animation / playback range.

Current Frame
: The current frame of the animation / playback range.
Also the position of the *Time Cursor*.

III Player Control

These button are used to set, play, rewind, the *Time Cursor*.


Player Controls.

Jump to start
: This sets the cursor to the start of frame range. ⇧ ShiftCtrlArrow down or ⇧ ShiftArrow left

Jump to previous keyframe
: This sets the cursor to the previous keyframe. Arrow down

Rewind
: This plays the animation sequence in reverse. ⇧ ShiftAltA
When playing the play buttons switch to a pause button.

Play
: This plays the animation sequence. AltA
When playing the play buttons switch to a pause button.

Jump to next keyframe
: This sets the cursor to the next keyframe. Arrow up

Jump to end
: This sets the cursor to the end of frame range. ⇧ ShiftCtrlArrow up or ⇧ ShiftArrow right

Pause
: This stops the animation. AltA

IV Synchronize Playback

3D View Red
FPS. 60:54.75

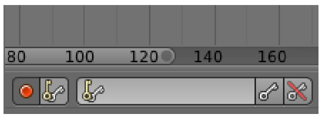
When you play an animation, the FPS is displayed at the top left of the 3D View.
If the scene is detailed and playback is slower than the set Frame Rate, these options are used to synchronize the playback.

No Sync
: Do not sync, play every frame.

Frame Dropping
: Drop frames if playback is too slow.
This enables *Frame Dropping* from the *Playback Menu*.

AV-sync
: Sync to audio clock, dropping frames if playback is slow.
This enables *AV-sync* and *Frame Dropping* from the *Playback Menu*.

V Keyframe Control



Timeline Auto Keyframe.

Auto Keyframe
: The "Record" red-dot button enables something called *Auto Keyframe*: It will add and/or replace existing keyframes for the

active object when you transform it in the 3D view.

For example, when enabled, first set the *Time Cursor* to the desired frame, then move an object in the 3d view, or set a new value for a property in the UI.

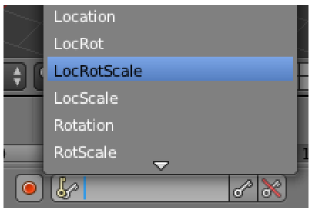
When you set a new value for the properties, blender will add keyframes on the current frame for the transform properties.

Auto Keying Set - Optional if Auto Keyframe enabled.
    Auto Keyframe will insert new keyframes for the properties in the active *Keying Set*.

Note that *Auto Keyframe* only works for transform properties (objects and bones), in the 3D views (i.e. you cant use it e.g. to animate the colors of a material in the Properties window…).

Timeline
Layered.

Layered - Optional while playback.
    Adds a new NLA Track + Strip for every loop/pass made over the animation to allow non-destructive tweaking.



Timeline Keying Sets.

Active Keying Set
    Keying Sets are a set of keyframe channels in one.

    They are made so the user can record multiple properties at the same time.

    With a keying set selected, when you insert a keyframe, blender will add keyframes for the properties in the active *Keying Set*.

    There are some built in keying sets, 'LocRotScale', and also custom keying sets.

    Custom keying sets can be defined in the in the panels *Properties > Scene > Keying Sets + Active Keying Set*.

Insert Keyframes
    Insert keyframes on the current frame for the properties in the active *Keying Set*.

Delete Keyframes
    Delete keyframes on the current frame for the properties in the active *Keying Set*.

User Preferences

Some related user preferences from the **Editing** tab.

Playback

AllowNegative Frames
    Time Cursor can be set to negative frames with mouse or keyboard.
    When using *Use PreviewRange*, this also allows playback.

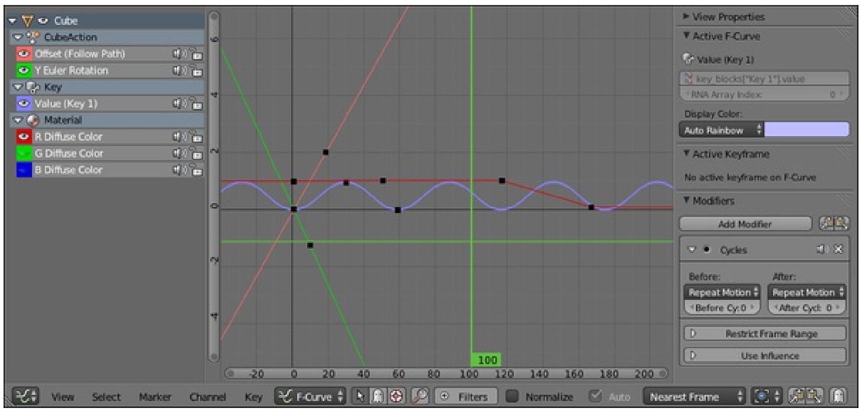
Keyframing

Visual Keying
    When an object is using constraints, the objects property value doesnt actually change.
    Visual Keying will add keyframes to the object property, with a value based on the visual transformation from the constraint.
Only Insert Needed
    This will only insert keyframes if the value of the propery is different.
Auto Keyframing
    Enable *Auto Keyframe* by default for new scenes.
ShowAuto Keying Warning
    Displays a warning at the top right of the *3D View*, when moving objects, if *Auto Keyframe* is on.
Only Insert Available
    With *Auto Keyframe* enabled, this will only add keyframes to channel F-Curves that already exist.

Graph Editor

The graph editor is the main animation editor. It allows you to modify the animation for any properties using *F-Curves*.

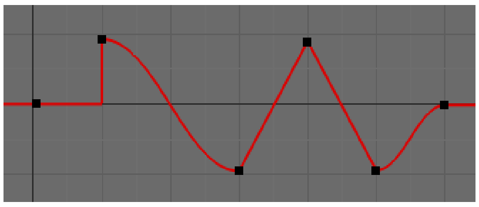
The graph editor has two modes, *F-Curve* for *Actions*, and *Drivers* for *Drivers*. Both are very similar in function.



The Graph Editor.

Curve Editor Area

Here you can see and edit the curves and keyframes.


A curve with different types of interpolation.

See *F-Curves* and *Editing* for more info.

Navigation

As with most windows, you can:

Pan MMB
  Pan the view vertically (values) or horizontally (time) with click and drag.

Zoom Wheel
  Zoom in and out with the mouse wheel.

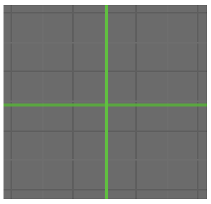
Scale View Ctrl MMB
  Scale the view vertically or horizontally.

These are some other useful tools.

View All ↖ Home
  Reset viewable area to show all keyframes.

View Selected Numpad.
  Reset viewable area to show selected keyframes.

2D Cursor


Graph Editor 2D Cursor.

The current frame is represented by a green vertical line called the *Time Cursor*.

As in the Timeline, you can change the current frame by pressing or holding LMB.

The green horizontal line is called the *Cursor*. This can be disabled via the *View Menu* or the *View Properties* panel.

The *Time Cursor* and the *Cursor* make the *2D Cursor*. The *2D Cursor* mostly used for editing tools.

View Axes

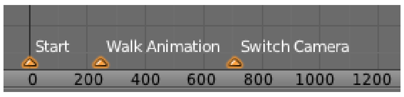
For *Actions* the X-axis represents time, the Y-axis represents the value to set the property.

For *Drivers* the X-axis represents the *Driver Value*, the Y-axis represents the value to set the property.

Depending on the selected curves, the values have different meaning: For example rotation properties are shown in degrees, location properties are shown in Blender Units. Note that *Drivers* use radians for rotation properties.

Markers

Like with most animation editors, markers are shown at the bottom of the editor.



Graph Editor Markers.

Markers can be modified in the *Graph Editor* though its usually best to use the *Timeline*.

See *Marker Menu* or *Markers* for more info.

Header

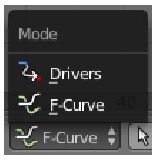
Here you'll find.

- The menus.
- Graph Editor mode.
- View controls.
- Curve controls.

Menus

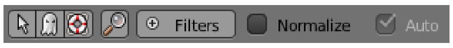
See *Header Menus* for more info.

Header Controls



Graph Mode.

Mode
: F-Curve for *Actions*, and Drivers for *Drivers*.



View Controls.

View controls

Show Only Selected
: Only include curves related to the selected objects and data.

Show Hidden
: Include curves from objects/bones that are not visible.

Show Only Errors
: Only include curves that are disabled or have errors.

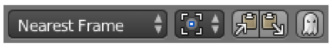
Search Filter
: Only include curves with keywords contained in the search text.

Type Filter
: Filter curves by property type.

Normalize
: Normalize curves so the maximum or minimum point equals 1.0 or -1.0.

Auto

Automatically recalculate curve normalization on every curve edit.



Curve Controls.

Curve controls

Auto Snap

Auto snap the keyframes for transformations.

No Auto-Snap
Time Step
Nearest Frame
Nearest Marker

Pivot Point

Pivot point for rotation.

Bounding Box Center

Center of the select keyframes.

2D Cursor

Center of the *2D Cursor*. *Time Cursor* + *Cursor*.

Individual Centers

Rotate the selected keyframe *Bezier* handles.

Copy Keyframes CtrlC

Copy the selected keyframes to memory.

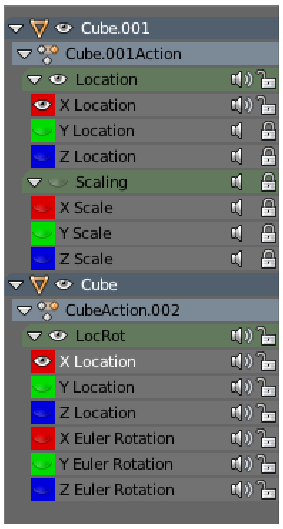
Paste Keyframes CtrlV

Paste keyframes from memory to the current frame for selected curves.

Create Snapshot

Creates a picture with the current shape of the curves.

Channels Region



Channels Region.

The channels region is used to select and manage the curves for the graph editor.

Hide curve
Represented by the eye icon.

Deactive/Mute curve
Represented by the speaker icon.

Lock curve from editing
Represented by the padlock icon.

Channel Editing

Select channel LMB

Multi Select/Deselect ⇧ Shift LMB

Toggle Select All A

Border Select Drag LMB or B Drag LMB

Border Deselect ⇧ ShiftDrag LMB or B ⇧ ShiftDrag LMB

Delete selected X or Delete

Lock selected ⇆ Tab

Make only selected visible V

Enable Mute Lock selected ⇧ ShiftCtrlW

Disable Mute Lock selected AltW

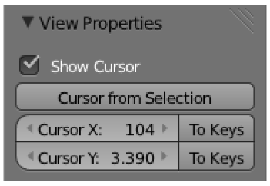
Toggle Mute Lock selected ⇧ ShiftW

See *Channel Menu* for more info.

Properties Region

The panels in the *Properties Region*.

View Properties Panel



View Properties Panel.

ShowCursor
: Show the vertical *Cursor*.

Cursor from Selection
: Set the *2D cursor* to the center of the selected keyframes.

Cursor X
: *Time Cursor* X position.

: *To Keys*

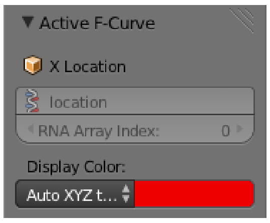
: : Snap selected keyframes to the *Time Cursor*.

Cursor Y
: Vertical *Cursor* Y position.

: *To Keys*

: : Snap selected keyframes to the *Cursor*.

Active F-Curve Panel



Active F-Curve Panel.

This panel displays properties for the active *F-Curve*.

Channel Name (X Location)
: *ID Type* + Channel name.

RNA Path

RNA Path to property + Array index.

Color Mode
: *Color Mode* for the active *F-Curve*.

  Auto Rainbow

    Increment the *HUE* of the *F-Curve* color based on the channel index.

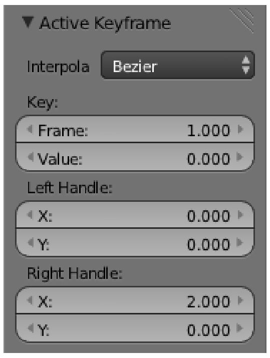
  Auto XYZ to RGB

    For property sets like location xyz, automatically set the set of colors to red, green, blue.

  User Defined

    Define a custom color for the active *F-Curve*.

Active Keyframe Panel


Active Keyframe
Interpola: Bezier
Key:
Frame: 1.000
Value: 0.000
Left Handle:
X: 0.000
Y: 0.000
Right Handle:
X: 2.000
Y: 0.000

Active Keyframe Panel.

Interpolation
: Set the forward interpolation for the active keyframe.

  Constant

    Keep the same value till the next keyframe.

  Linear

    The difference between the next keyframe.

  Bezier

    Bezier interpolation to the next keyframe.

About additional interpolation modes see here.

Key
: *Frame*

    Set the frame for the active keyframe.

  Value

    Set the value for the active keyframe.

Specific Bezier interpolation controls
Left Handle
: Set the position of the left interpolation handle for the active keyframe.

Right Handle
: Set the position of the right interpolation handle for the active keyframe.

Specific Easing (by strength) and Dynamic Effects interpolation controls
Easing

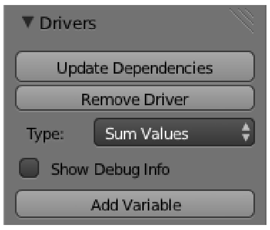
  Automatic Easing
  Ease In
  Ease Out
  Ease In and Out

  About methods of interpolation easing see here.

Back
: Amount of overshoot for «*Back*» easing

Amplitude
Amount to boost elastic bounces for «*Elastic*» easing
Period
Time between bounces for «*Elastic*» easing.

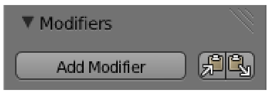
Drivers Panel



Drivers Panel.

See Drivers Panel for more info.

Modifiers Panel



Modifiers Panel.

See F-Modifiers for more info.

See Also

- Graph Editor - F-Curves
- Graph Editor - F-Modifiers
- Graph Editor - Editing
- Actions
- Drivers

F-Curves

Once you have created keyframes for something, you can edit their corresponding curves. In Blender 2.5, IPO Curves have been replaced by F-Curves, however, editing these curves is essentially still the same.

The concept of Interpolation

When something is "animated," it changes over time. In Blender, animating an object means changing one of its properties, such as its X location, or the Red channel value of its material diffuse color, and so on, during a certain amount of time.

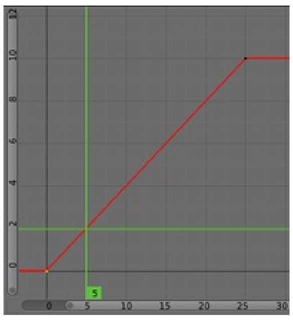
As mentioned, Blender's fundamental unit of time is the "frame", which usually lasts just a fraction of a second, depending on the *frame rate* of the scene.

As animation is composed of incremental changes spanning multiple frames, usually these properties ARE NOT manually modified *frame by frame*, because:

- it would take ages!
- it would be very difficult to get smooth variations of the property (unless you compute mathematical functions and type a precise value for each frame, which would be crazy).

This is why nearly all direct animation is done using **interpolation**.

The idea is simple: you define a few Key Frames, which are multiple frames apart. Between these keyframes, the properties' values are computed (interpolated) by Blender and filled in. Thus, the animators' workload is significantly reduced.


Example of interpolation

For example, if you have:

- a control point of value **0** at frame **0**,
- another one of value **10** at frame **25**,
- linear interpolation,

then, at frame **5** we get a value of **2**.

The same goes for all intermediate frames: with just two points, you get a smooth growth from **0** to **10** along the **25 frames**. Obviously, if you'd like the frame **15** to have a value of **9**, you'd have to add another control point (or keyframe)…

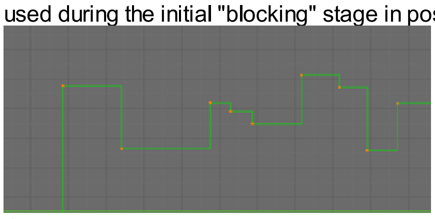
Settings

F-curves have three additional properties, which control the interpolation between points, extension behavior, and the type of handles.

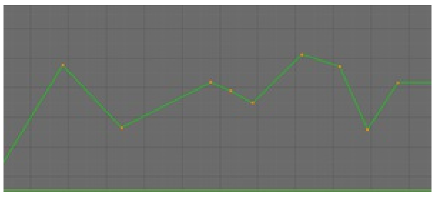
Interpolation Mode

You have three choices (T, or Key » Interpolation Mode):

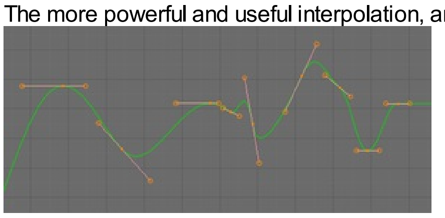
Constant
: There is no interpolation at all. The curve holds the value of its last keyframe, giving a discrete (stairway) "curve". Usually only used during the initial "blocking" stage in pose-to-pose animation workflows.


Constant.

Linear
: This simple interpolation creates a straight segment between each neighbor keyframes, giving a broken line. It can be useful when using only two keyframes and the Extrapolation extend mode, to easily get an infinite straight line (i.e. a linear curve).

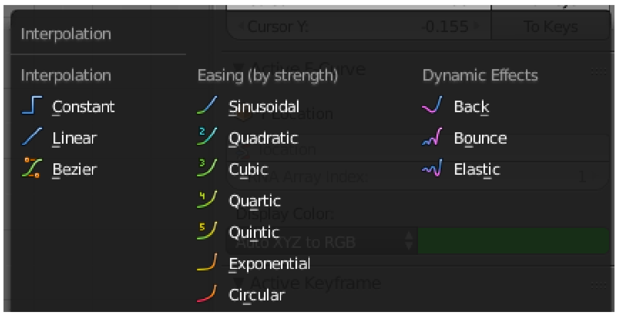

Linear.

Bezier
The more powerful and useful interpolation, and the default one. It gives nicely smoothed curves, i.e. smooth animations!


Bézier.

Remember that some Fcurves can only take discrete values, in which case they are always shown as if constant interpolated, whatever option you chose.

Additional interpolation modes



Available Interpolation Modes

Also now there are availabled additional interpolation modes:

Easing (by strenght)
The different methods of easing intrpolations for F-Curve segment.

Sinosuidal
Sinusoidal easing (weakest, almost linear but with a slight curvature).
Quadratic
Quadratic easing.
Cubic
Cubic easing.
Quartic
Quartic easing.
Quintic
Quintic easing.
Exponential
Exponential easing (dramatic)
Circular
Circular easing (strongest and most dynamic)

Dynamic Effects
Some dynamic interpolations between two keyframes on F-Curve.

Back
Cubic easing with overshoot and settle.
Bounce
Exponentially decaying parabolic bounce, like when objects collide.
Elastic
Exponentially decaying sine wave, like an elastic band.

Easing
The way in which ends of the segment between this and the next keyframe easing interpolation is applied to

Automatic Easing
Easing type is chosen automatically based on what the type of interpolation used (e.g. 'Ease In' for transitional types, and 'Ease Out' for dynamic effects).
Ease In

Only on the end closest to the next keyframe.

Ease Out
: Only on the end closest to the first keyframe.

Ease In and Out
: Segment between both keyframes.

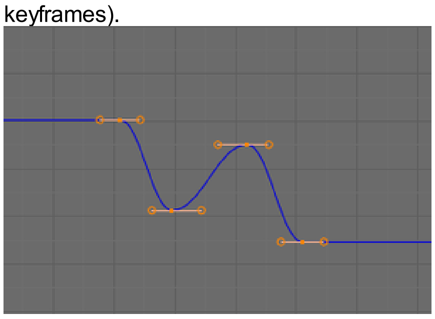
Extrapolation

(⇧ ShiftE, or Channel » Extrapolation Mode)

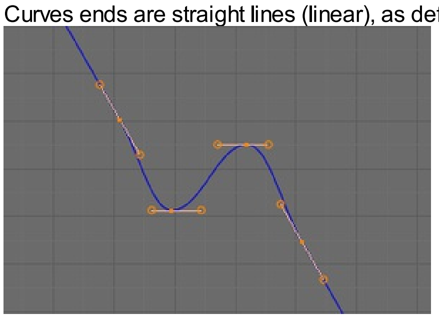
Extrapolation defines the behavior of a curve before the first and after the last keyframes.

There are two basic extrapolation modes:

Constant
: The default one, curves before their first keyframe and after their last one have a constant value (the one of these first and last keyframes).


Constant extrapolation

Linear
: Curves ends are straight lines (linear), as defined by their first two keyframes (respectively their last two keyframes).

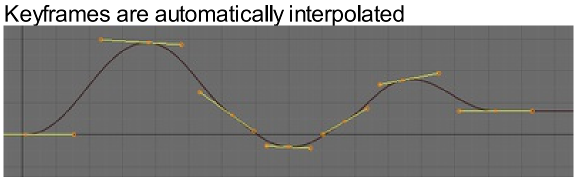

Linear extrapolation

Additional extrapolation tools (e.g. the "Cycles" F-Modifier) are located in the F-Curve Modifiers

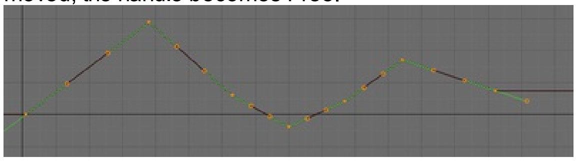
Handle Types

There is another curve option quite useful for Bézier-interpolated curves. You can set the type of handle to use for the curve points V

Automatic
: Keyframes are automatically interpolated

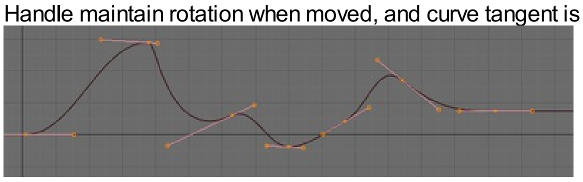

Auto handles

Vector
: Creates linear interpolation between keyframes. The linear segments remain if keyframe centers are moved. If handles are moved, the handle becomes Free.


Vector handles

Aligned

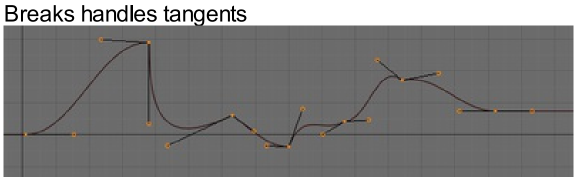
Handle maintain rotation when moved, and curve tangent is maintained



Aligned handles

Free

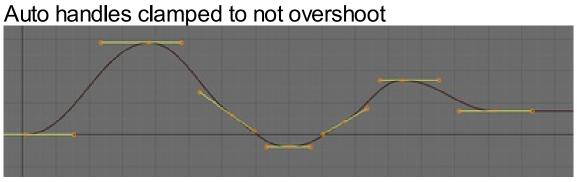
Breaks handles tangents



Free handles

Auto Clamped

Auto handles clamped to not overshoot



Auto clamped handles

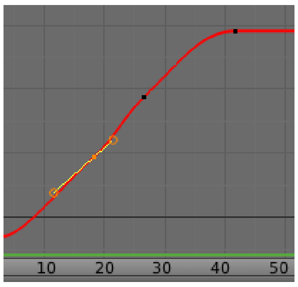
Direction of time

Although F-curves are very similar to Bézier curves, there are some important differences.

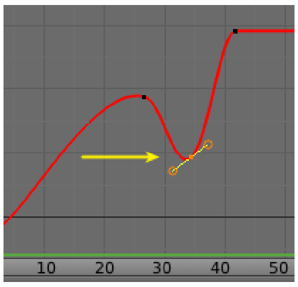
For obvious reasons, **a property represented by a Curve cannot have more than one value at a given time**, hence:

- when you move a control point ahead of a control point that was previously ahead of the point that you are moving, the two control points switch their order in the edited curve, to avoid that the curve goes back in time
- for the above reason, it's impossible to have a closed Ipo curve

Two control points switching: the curve can't go back in time!



Before moving the second keyframe

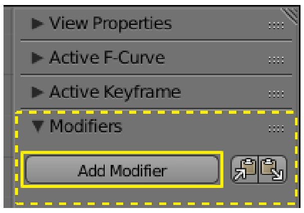


After moving the second keyframe

F-Curve Modifiers

F-Curve modifiers are similar to object modifiers, in that they add non-destructive effects, that can be adjusted at any time, and layered to create more complex effects.

Adding a Modifier



The F-curve modifier panel.

The F-curve modifier panel is located in the Properties panel. Select a curve by selecting one of its curve points, or by selecting the channel list. Click on the Add Modifier button and select a modifier.

To add spin to an object or group, select the object/group and add a keyframe to the axis of rotation (x,y, or z)

Go to the Graph Editor.....make sure the f-curves properties panel is visible (View > Properties)

>Add Modifier > (e.g.) Generator

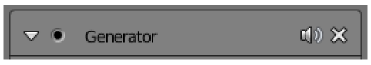
Copy F-Modifiers button
Copy the F-Modifier(s) of the active F-Curve.

Paste F-Modifiers button
Adds copied F-Modifiers to the selected F-Curves.

Common Interface

All of modifiers have common the top and bottom parts of their panel.

Top (from left to right):



Common top interface elements for all modifiers.

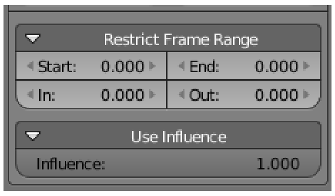
A small arrow
This control allows you to show/hide the modifier settings.

The modifier type
This is just static text showing you what this modifier is…

Mute modifier button
Enables/disables modifier evaluation for the F-curve.

The “X” control
This will remove from the Modifiers panel the modifier.

Bottom (from up to down):



Common bottom interface elements for all modifiers.

Restrict Frame Range
F-Curve Modifier is only applied for the specified frame range to help mask off effects in order to chain them.

Start
Frame that modifier’s influence starts (if Restrict Frame Range is in use).
End
Frame that modifier’s influence ends (if Restrict Frame Range is in use).
In

Number of frames from start frame for influence to take effect.

Out

Number of frames from end frame for influence to fade out.

Use Influence

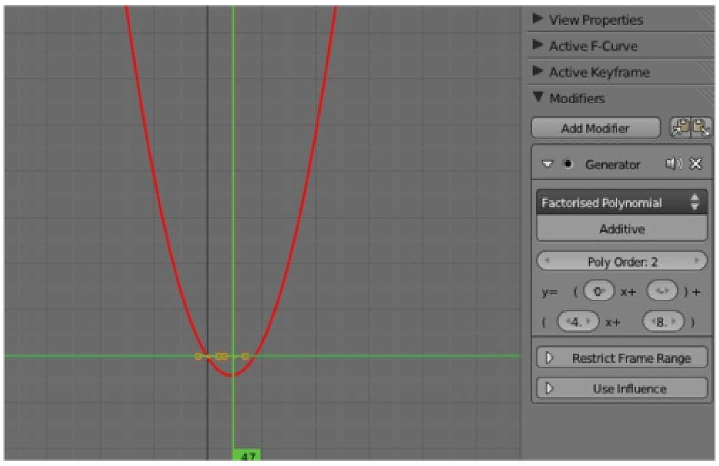
F-Curve Modifier’s effects will be tempered by a default factor.

Influence

Amount of influence F-Curve Modifier will have when not fading in/out.

Types of Modifiers

Generator



The Generator modifier.

Generator creates a Factorized or Expanded Polynomial function. These are basic mathematical formulas that represent lines, parabolas, and other more complex curves, depending on the values used.

Additive

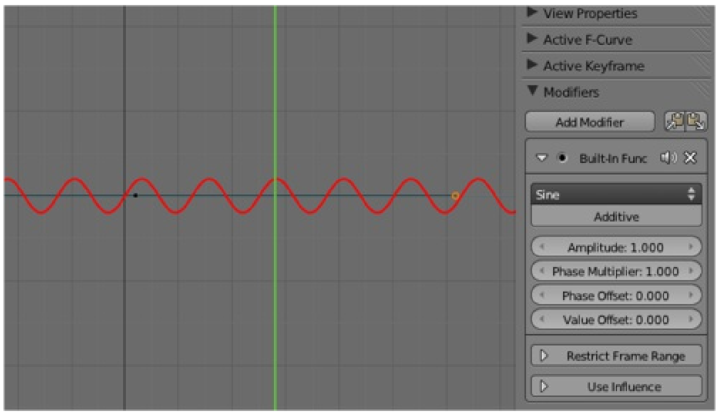
This option causes the modifier to be added to the curve, instead of replacing it by default.

Poly Order

Specify the order of the polynomial, or the highest power of ‘x’ for this polynomial. (number of coefficients - 1).

Change the Coefficient values to change the shape of the curve. See The Wikipedia Page for more information on polynomials.

Built-in Function



The Built-in Function modifier.

These are additional formulas, each with the same options to control their shape. Consult mathematics reference for more detailed information on each function.

- Sine
- Cosine
- Tangent
- Square Root
- Natural Logarithm
- Normalized Sine (sin(x)/x)

Amplitude

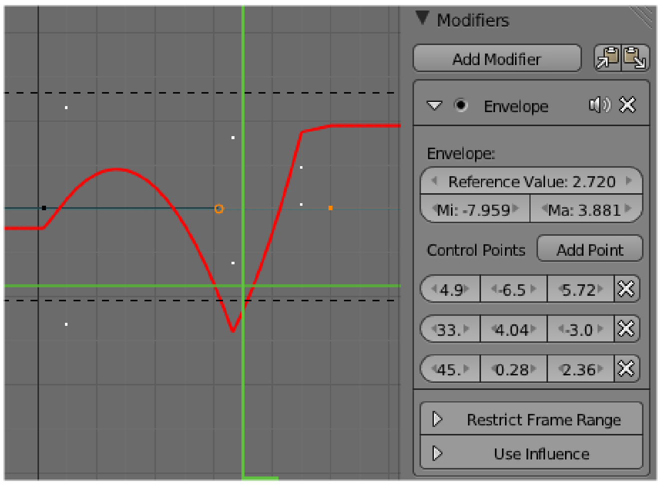
Adjusts the Y scaling

Phase Multiplier

Adjusts the X scaling

Phase Offset
    Adjusts the X offset
Value Offset
    Adjusts the Y offset

Envelope



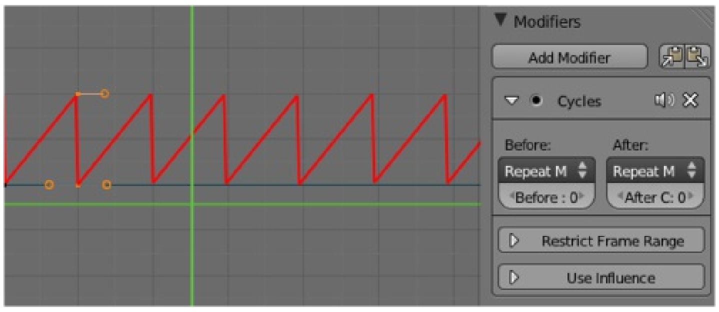
The Envelope modifier.

Allows you to adjust the overall shape of a curve with control points.

Reference Value
    Set the Y value the envelope is centered around.
Min
    Lower distance from Reference Value for 1:1 default influence.
Max
    Upper distance from Reference Value for 1:1 default influence.

Add Point
    Add a set of control points. They will be created at the current frame.
Fra:
    Set the frame number for the control point.
Min
    Specifies the lower control point's position.
Max
    specifies the upper control point's position.

Cycles



The Cycles modifier.

Cycles allows you add cyclic motion to a curve that has 2 or more control points. The options can be set for before and after the curve.

Cycle Mode
    Repeat Motion

        Repeats the curve data, while maintaining their values each cycle.

    Repeat with Offset

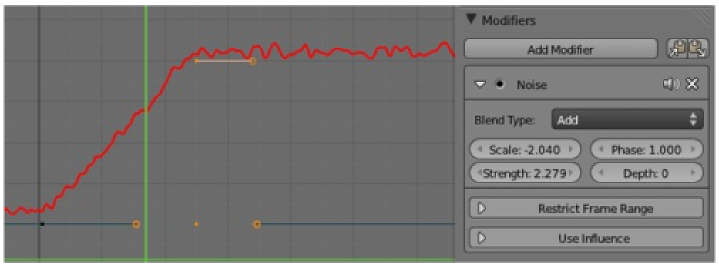
        Repeats the curve data, but offsets the value of the first point to the value of the last point each cycle.

    Repeat Mirrored

        Each cycle the curve data is flipped across the X-axis.

Before/After Cycles
Set the number of times to cycle the data. A value of 0 cycles the data infinitely.

Noise



Blend Type: Add
Scale: -2.040
Phase: 1.000
Strength: 2.279
Depth: 0

The Noise modifier.

Modifies the curve with a noise formula. This is useful for creating subtle or extreme randomness to animated movements, like camera shake.

Blend Type
Replace

Adds a -.5 to .5 range noise function to the curve.

Add

Adds a 0 to 1 range noise function to the curve.

Subtract

Subtracts a 0 to 1 range noise function to the curve.

Multiply

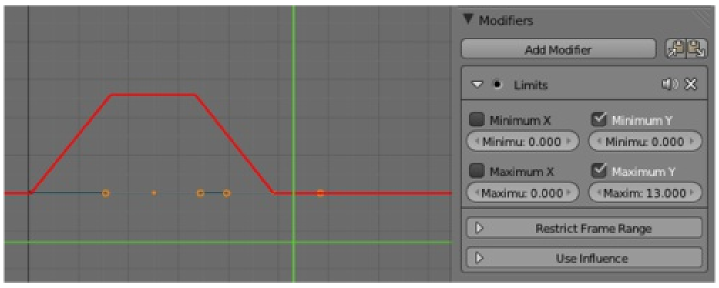
Multiplies a 0 to 1 range noise function to the curve.

Scale
Adjust the overall size of the noise. Values further from 0 give less frequent noise.
Strength
Adjusts the Y scaling of the noise function.
Offset
Time offset for the noise effect.
Phase
Adjusts the random seed of the noise.
Depth
Adjusts how detailed the noise function is.

Python

This can use Python script to generate values over time that are relayed to the parameter they are assigned to. It is mostly using with working F-curves in Drivers mode.

Limits



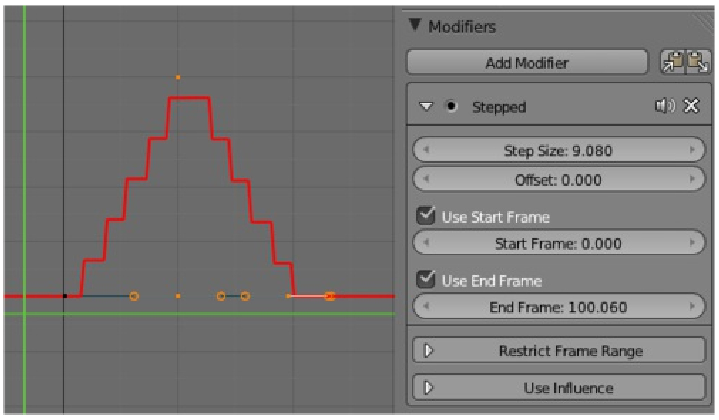
Minimum X
Minimum Y ✓
Minimu: 0.000
Minimu: 0.000
Maximum X
Maximum Y ✓
Maximu: 0.000
Maxim: 13.000

The Limits modifier.

Limit curve values to specified X and Y ranges.

Minimum/Maximum X
Cuts a curve off at these frames ranges, and sets their minimum value at those points.
Minimum/Maximum Y
Truncates the curve values to a range.

Stepped Interpolation



The Stepped Interpolation modifier.

Gives the curve a stepped appearance by rounding values down within a certain range of frames.

Step Size
: Specify the number of frames to hold each frame

Offset
: Reference number of frames before frames get held. Use to get hold for ‘1-3’ vs ‘5-7’ holding patterns.

Use Start Frame
: Restrict modifier to only act before its ‘end’ frame

Use End Frame
: Restrict modifier to only act after its ‘start’ frame

Graph Editor - Editing

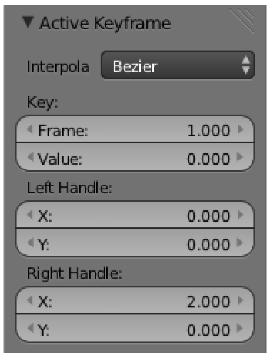
Tools and menus for the *F-Curves* in the *Graph Editor*.

By default, when new channels are added, the *Graph Editor* sets them to *Edit Mode*. Selected channels can be locked by pressing ⇆ Tab.

Basic Tools

These are some basic tools to modify the curves and keyframes.

Transformations



Interpola: Bezier
Key:
Frame: 1.000
Value: 0.000
Left Handle:
X: 0.000
Y: 0.000
Right Handle:
X: 2.000
Y: 0.000

Active Keyframe Panel.

Grab/Move selected keyframes G

Rotate selected keyframes R

Scale selected keyframes S

Additionally, for translation and scaling, you can lock the transformation along the X (time frame) or Y (value), as usual by pressing X or Y during transformation.

For precise control of the keyframe position and value, you can set values in the *Active Keyframe* of the Properties Region.

Selection

Select Keyframe RMB

Toggle Select multiple keyframes ⇧ Shift RMB

Toggle Select All A

Select Linked L

Border Select B Drag LMB

Border Deselect B ⇧ ShiftDrag LMB

Editing

Duplicate selected keyframes ⇧ ShiftD

Add keyframe to active curve Ctrl LMB

Insert keyframes to the Time Cursor I

Copy Keyframes CtrlC
: Copy the selected keyframes to memory.

Paste Keyframes CtrlV
: Paste keyframes from memory to the current frame for selected curves.

Curves and Keyframes

Set Keyframe Extrapolation ⇧ ShiftE
- *Constant Extrapolation*
- *Linear Extrapolation*
- *Make Cyclic (F-Modifier)*
- *Clear Cyclic (F-Modifier)*

Set Keyframe Interpolation T
- *Constant*
- *Linear*

Bezier

Set Keyframe Handle Type V
: *Free*
: *Vector*
: *Aligned*
: *Automatic*
: *Auto Clamped*

See F-Curves for more info.

View Tools

ViewAll ↖ Home

ViewSelected Numpad.

Set PreviewRange CtrlP

Auto-Set PreviewRange CtrlAltP

Clear PreviewRange AltP

Toggle ShowHandles CtrlH

Toggle ShowSeconds CtrlT

More Tools

Some other tools used to modify the the curves and keyframes.

Transform Snapping

When transforming keyframes with G, R, S, the transformation can be snapped to increments.

Snap Transformation to 1.0 Ctrl

Divide Transformation by 10.0 ⇧ Shift

Keyframes can be snapped to different properties by using the *Snap Keys* tool.

Snap Keys ⇧ ShiftS

Current Frame
: Snap the selected keyframes to the *Time Cursor*.

Cursor Value
: Snap the selected keyframes to the *Cursor*.

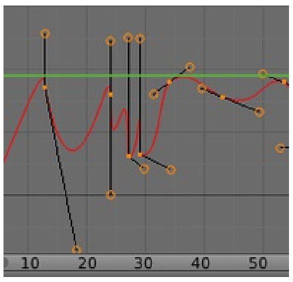
Nearest Frame
: Snap the selected keyframes to their nearest frame individually.

Nearest Second
: Snap the selected keyframes to their nearest second individually, based on the *FPS* of the scene.

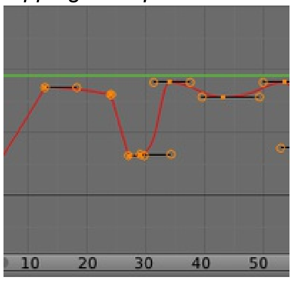
Nearest Marker
: Snap the selected keyframes to their nearest marker individually.

Flatten Handles
: Flatten the *Bezier* handles for the selected keyframes.

Flatten Handles snapping example.



Before Flatten Handles.



After Flatten Handles.

Mirror

Selected keyframes can be mirrored over different properties using the the *Mirror Keys* tool.

Mirror Keys ⇧ ShiftM
: By Times Over Current Frame

    Mirror horizontally over the *Time Cursor*.

  By Values over Cursor Value

    Mirror vertically over the *Cursor*.

  By Times over Time 0

    Mirror horizontally over frame 0.

  By Values over Value 0

    Mirror vertically over value 0.

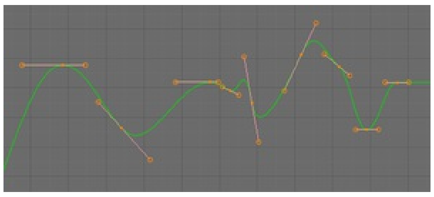
  By Times over First Selected Marker

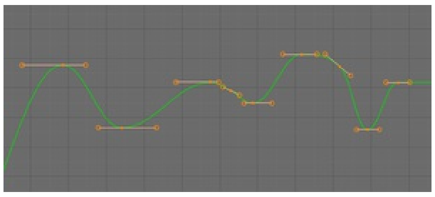
    Mirror horizontally the over the first selected *Marker*.

Clean Keyframes

Clean Keyframes resets the keyframe tangents to their auto-clamped shape, if they have been modified.

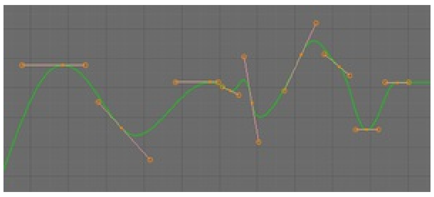
Clean Keyframes O

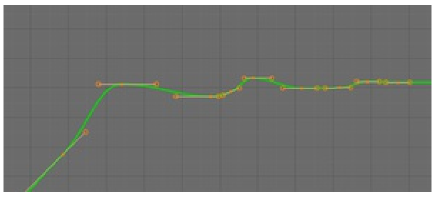

Fcurve before cleaning


Fcurve after cleaning

Smoothing

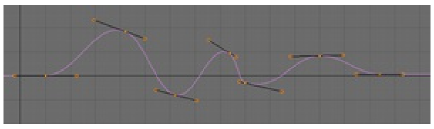
(AltO or Key » Smooth Keys) There is also an option to smooth the selected curves , but beware: its algorithm seems to be to divide by two the distance between each keyframe and the average linear value of the curve, without any setting, which gives quite a strong smoothing! Note that the first and last keys seem to be never modified by this tool.

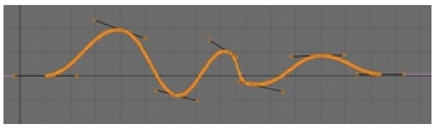

Fcurve before smoothing


Fcurve after smoothing

Sampling and Baking Keyframes

Sample Keyframes ⇧ ShiftO
: Sampling a set a keyframes replaces interpolated values with a new keyframe for each frame.


Fcurve before sampling


Fcurve after sampling

Bake Curves AltC
: Baking a curve replaces it with a set of sampled points, and removes the ability to edit the curve.

Header Menus

Graph Editor header menus.

View Menu

Apart from the standard options like zoom-in/out, maximize window, center view on cursor, etc., this menu gathers various other options.

Properties N
: Opens the properties panel on the right side of the graph editor.

Realtime Updates
: When transforming keyframes, changes to the animation data are flushed to other views.

Show Frame Number Indicator
: Show frame number beside the current frame indicator line.

Show Cursor
: Shows the 2d cursor.

Show Sliders
: Show sliders beside F-Curve channels.

Show Group Colors
: Draw groups and channels with colors matching their corresponding groups.

AutoMerge Keyframes
: Automatically merge nearby keyframes.

Use High Quality Drawing
: Draw F-Curves using Anti-Aliasing and other fancy effects (disable for better performance).

Show Handles
: Show handles of Bezier control points.

Only Selected Curve Keyframes
: Only keyframes of selected F-Curves are visible and editable.

Only Selected Keyframe handles
: Only show and edit handles of selected keyframes.

Show Seconds
: Show timing in seconds not frames.

Set Preview Range, Clear Preview Range (CtrlP, AltP)
: These entries allow you to define/clear a temporary preview range to use for the AltA realtime playback (this is the same thing as the Pr option of the Timeline window header).

Auto-Set Preview Range CtrlAltP
: Automatically set Preview Range based on range of keyframes.

View All ↖ Home
: Reset viewable area to show full keyframe range.

View Selected Pad.
: Reset viewable area to show selected keyframe range.

Select Menu

Select All A
: In edit mode, select/deselect all keyframes.
In locked mode, select/deselect all visible channels.

Invert Selection CtrlI
: Inverts selected keys.

Border Select B
: Allows selection of keyframes within a region.

Border Axis Range AltB
: Axis Range...

Border (include Handles CtrlB
: Include Handles, handles tested individually against the selection criteria.

Columns on Selected Keys K
: Select all keys on same frame as selected one(s).

Column on current Frame CtrlK
: Select all keyframes on the current frame.

Columns on selected Markers ⇧ ShiftK
: Select all keyframes on the frame of selected marker(s).

Between Selected Markers AltK
: Select all keyframes between selected markers.

Before Current Frame [
: Select all keys before the current frame.

After Current Frame]
: Select all keys after the current frame.

Select More Ctrl+ NumPad
: Grow keyframe selection along Fcurve.

Select Less Ctrl- NumPad
: Shrink keyframe selection along Fcurve.

Select Linked L
: Selects all keyframes on Fcurve of selected keyframe.

Marker Menu

Add Marker M

Duplicate Marker ⇧ ShiftD

Duplicate Marker to Scene

Delete Marker X or Delete
: Note, make sure no channels are selected.

Rename Marker CtrlM

Grab/Move Marker Tweak select

Jump to Next Marker

Jump to Previous Marker

Channel Menu

Delete Channels X or Delete

Group Channels CtrlG

Ungroup Channels AltG

Toggle Channel Settings ⇧ ShiftW
: *Protect*
: *Mute*

Enable Channel Settings ⇧ ShiftCtrlW
: *Protect*
: *Mute*

Disable Channel Settings AltW
: *Protect*
: *Mute*

Toggle Channel Editability ⇆ Tab

Set Visibilty V

Extrapolation Mode ⇧ ShiftE
: *Constant Extrapolation*
: *Linear Extrapolation*
: *Make Cyclic (F-Modifiers)*
: *Clear Cyclic (F-Modifiers)*

Expand Channels Numpad+

Collapse Channels Numpad-

Move...
: *To Top* ⇧ ShiftPageup
: *Up* Pageup
: *Down* Pagedown
: *To Bottom* ⇧ ShiftPagedown

Revive Disabled F-Curves

Key Menu

Transform
: *Grab/Move* G
: *Extend* E

- *Rotate* R
- *Scale* S

Snap ⇧ ShiftS
- *Current Frame*
- *Cursor Value*
- *Nearest Frame*
- *Nearest Second*
- *Nearest Marker*
- *Flatten Handles*

Mirror ⇧ ShiftM
- *By Times over Current Frame*
- *By Values over Current Value*
- *By Times over Time=0*
- *By Values over Value=0*
- *By Times over First Selected Marker*

Insert Keyframes I

Add F-Curve Modifier

Bake Sound to F-Curves

Jump to Keyframes CtrlG

Duplicate ⇧ ShiftD

Delete Keyframes X or Delete

Handle Type V
- *Free*
- *Vector*
- *Aligned*
- *Automatic*
- *Auto Clamped*

Interpolation Mode T
- *Constant*
- *Linear*
- *Bezier*

Clean Keyframes O

Smooth Keyframes AltO

Sample Keyframes ⇧ ShiftO

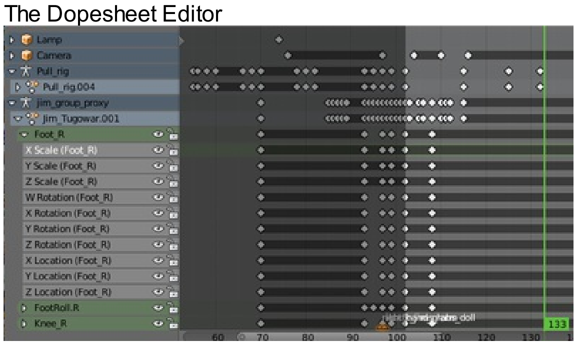
Bake Curve AltC

Copy Keyframes CtrlC

Paste Keyframes CtrlV

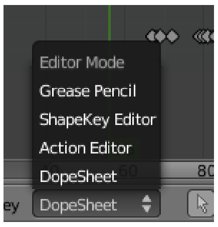
Discontinuity (Euler) Filter

The Dopesheet Editor


The DopeSheet editor

Classical hand-drawn animators often made a chart, showing exactly when each drawing, sound and camera move would occur, and for how long. They nicknamed this the 'dopesheet'. While CG foundations dramatically differ from classical hand-drawn animation, Blender's Dopesheet inherits a similar directive. It gives the animator a 'birds-eye-view' of every thing occurring within a scene.

Dope Sheet Modes



DopeSheet modes

There are five basic views for the Dopesheet. These all view different contexts of animation:

DopeSheet
: The dopeSheet allow you to edit multiple actions at once.

Action Editor
: Action Editor is the default, and most useful one. It’s here you can define and control your actions.

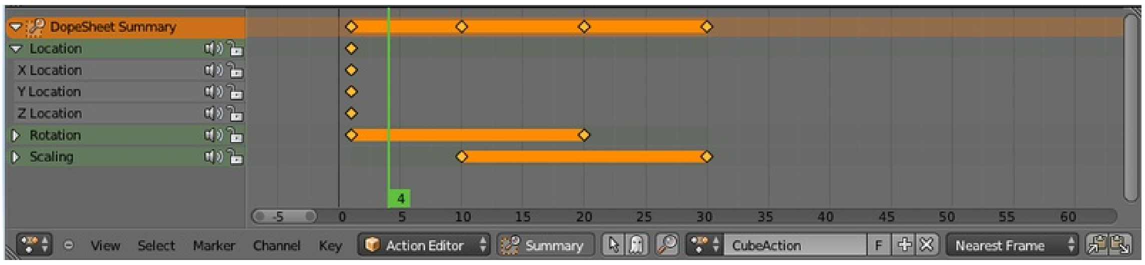
Shape Key Editor
: ShapeKey Editor is dedicated to the Shape Ipo datablocks. It uses/edits the same action datablock as the previous mode. It seems to be an old and useless thing, as the Action Editor mode handles Shape channels very well, and this mode adds nothing…

Grease Pencil
: Grease Pencil is dedicated to the grease pencil tool’s keyframes – for each grease pencil layer, you have a strip along which you can grab its keys, and hence easily re-time your animated sketches. As it is just another way to see and edit the grease pencil data, this mode uses no datablock (and hence has nothing to do with actions…). Note that you’ll have as much top-level grease pencil channels as you have sketched windows (3D views, UV/Image Editor, etc.)

Mask
: This mode is using for working with animation timing of masks from an external footages. See more details about the Mask mode here and a masks — here.

Interface

The Dope Sheet interface is similar to the Graph Editor one, it is divided in three areas:



The Dope Sheet Editor window, Action Editor mode, with an Object and Shape channels.

Header

Here you find the menus, a first block of controls related to the editor “mode”, a second one concerning the action datablocks, and a few other tools (like the copy/paste buttons, and snapping type).

The *Header* of Dope Sheet Editor almost identical to the Graph Editor one, except:

- *Pivots* selector for keyframes rotation/scaling (or more exactly their handles in the Bezier interpolation mode);
- *Snapshot Ghost* button of F-curves.

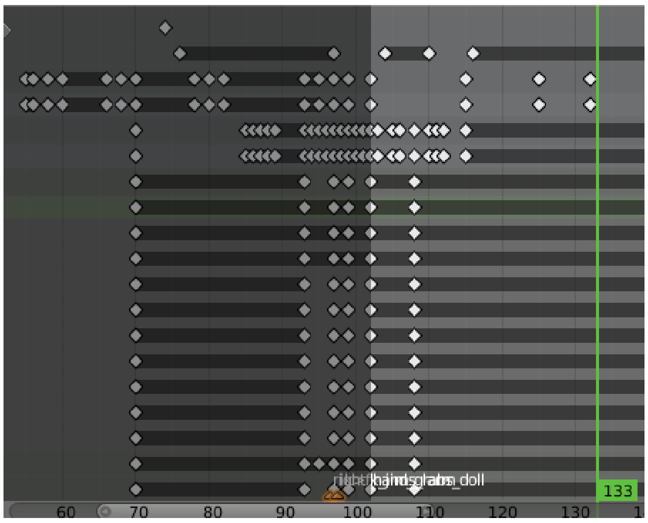
Also, the header of Dope Sheet Editor has the different sets of menus for their modes:

- Shape Key editor adn Mask modes has not the menu Channel;
- Grease Pencil mode — the menu Key is cliped to menu Frame with only duplicate and transform options.

By their functionality, menus of the *Header* of Dope Sheet Editor is a same as menus of the Graph Editor and Timeline window. See more info about:

- Graph Editor
- FCurve Editing
- Timeline
- Markers
- Actions

Edit Area



Edit area of the DopeSheet editor

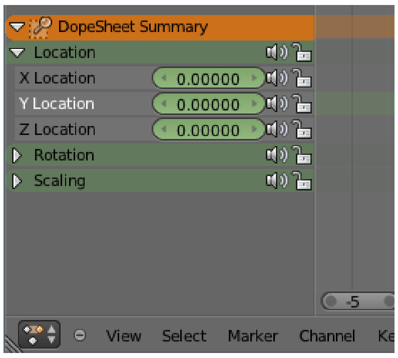
It contains the keyframes for all visible action channels.

As with the other “time” windows, the X-axis materializes the time. The Y-axis has no mean in itself, unlike with the Graph Editor, it’s just a sort of “stack” of action channels – each one being shown as an horizontal colored strip (of a darker shade “during” the animated/keyed period).

On these channel strips lay the keyframes, materialized as light-gray (unselected) or yellow (selected) diamonds.

One of the key feature of this window is that it allow you to visualize immediately which channel (i.e. Ipo curve) is *really* affected. When the value of a given channel does not change at all between two neighboring keyframes, a gray (unselected) or yellow (selected) line is drawn between them.

Channel Region



The Channel Region of Dope Sheet Editor

This is left “list-tree” part of the Dope Sheet Editor which shows the action’s channel “headers” and their hierarchy. Basically, there are:

- “Top-level” channels, which represent whole FCurve datablocks (so there’s one for Object one, one for Shape one, etc.). They gather *all* keyframes defined in their underlying FCurve datablock.
- “Mid-level” channels, which seem currently to have no use (there’s one per top-level channel, they are all named FCurves, and have no option at all…).
- “Low-level” channels, which represent individual FCurve , with their own keyframes (fortunately, only keyframes are shown!).

Each level can be expended/collapsed by the small arrow to the left of its “parent” channel.

To the right of the channel’s headers, there are some channel’s setting controls:

- Clicking on the small “speaker” will allow you to mute that channel.
- Clicking on the small “lock” will allow you to prevent this channel.

A channel can be selected (text in white) or not (text in black), use LMB clicks to toggle this state or ⇧ Shift LMB to multiply selection.

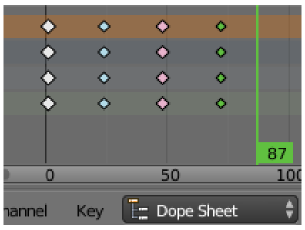
You can rename “Top-level” and “Mid-level” channels by twice clicking LMB on its header.

Keyframe Type

In «pose-to-pose» animation all of an animation events or states is divided by the significance on different pose or states. See more info about this here.

For visually distinguish different poses (extremes, breakdowns, other inbetweens) and regular keyframes in the Dopesheet editor there is the possibility the applying different colors for them. It is represented by feature «Keyframe Types».

All it do is change the color. The main intent is to help visual sorting when you are looking at your Dope Sheet.



The Keyframe Types (from left to rigth): *Keyframe*, *Breakdown*, *Extreme* and *Jitter*, in edit area of the DopeSheet editor

To select the required keyframe type there is available submenu Keyframe Type in Key menu of header Dope Sheet editor:

Keyframe
: Normal keyframe - e.g. for key poses.

Breakdown
: A breakdown pose - e.g. for transitions between key poses.

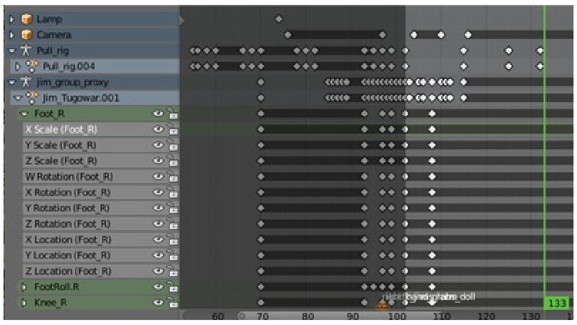
Extreme
: An ‘extreme’ pose, or some other purpose as needed.

Jitter
: A filler or baked keyframe for keying on ones, or some other purpose as needed.

Dope Sheet

Dope Sheet mode of Dope Sheet Editor is main mode of this editor and allows you to edit multiple actions at once.

Interface



The DopeSheet

Interface of the Dope Sheet mode is universal as for whole Dope Sheet Editor and about these you can see more details in corresponding sections:

- Dope Sheet Editor
- Graph Editor
- FCurve Editing
- Timeline
- Markers
- Actions

Action Editor

Blender 2.5 simplifies the system by making Actions the generic containers for F-Curves. Actions can contain any number of F-Curves, and can be attached to any data block.As long as the RNA data paths stored in the Action's F-Curves can be found on that data block, the animation will work. For example, an action modifying 'X location' and 'Y location' properties can be shared across multiple objects, since both objects have 'X location' and 'Y location' properties beneath them.

The Action Editor window enables you to see and edit the FCurve datablocks you defined as actions in the FCurve Editor window. So it takes place somewhere in-between the low-level FCurves, and the high-level NLA editor. Hence, you do not have to use them for simple Ipo curves animations – and they have not much interest in themselves, so you will mostly use this window when you do NLA animation (they do have a few specific usages on their own, though, like e.g. with the Action constraint, or the pose libraries…).

This is not a mandatory window, as you do can edit the actions used by the NLA directly in the FCurve Editor window (or even the NLA Editor one). However, it gives you a slightly simplified view of your FCurve datablocks (somewhat similar to the “key” mode of the FCurve window, even though more powerful in some ways) – and, more interesting, it can show you all ”action” FCurve datablocks of a same object at once.

Additionally, it also allows you to affect timing of the different keys of the layers created with the grease pencil tool.

Each “action” FCurve datablock forms a top-level channel (see below). Note that an object can have several Constraint (one per animated constraint) and Pose (for armatures, one per animated bone) FCurve datablocks, and hence an action can have several of these channels.

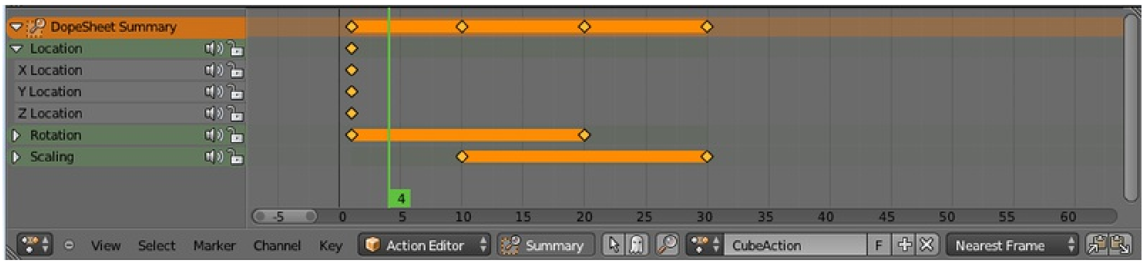
Action Datablocks

As everything else in Blender, actions are datablocks. Unlike FCurve ones, there is only one type of action, which can regroup all FCurve of a given object. You’ll find its usual datablock controls in the Action Editor header.

However, there is one specificity with action datablocks: they have by default a “fake user”, i.e. once created, they are always kept in Blender file, even if no object uses them. This is due to the fact that actions are designed to be used in the NLA, where you can affect several different actions to a same object! Yes, this is the only way to use different actions (and hence, different FCurve datablocks of the same kind) to animate a same object. But as you have to assign an action to an object to be able to edit it (and an object can only have one action datablock at a time), to have “fake users” guaranties you that you won’t lost your precious previously-edited actions when you start working on a new one!

This window shows, by default, the action datablock linked to the current active object. However, as with FCurvs, you can pin an Action Editor to a given action with the small “pin” button to the left of the datablock controls, in the header. This will force the window to always display this datablock, whatever the current selected object is.

Interface



The Action Editor mode of the Dope Sheet Editor window.

By their functionality, header, menus and window interface of the Action Editor is universal as for whole Dope Sheet Editor and about these you can see more details in corresponding sections:

- Dope Sheet Editor
- Graph Editor
- FCurve Editing
- Timeline
- Markers
- Actions

Using Action Editor

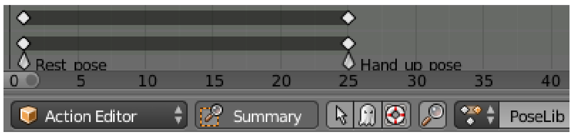
You may set and adjust a same set of animation events in Graph Editor with editing F-curves.

Then, you can to group these action F-curves to single action.

Thereafter, you add constraint Action to the required object or armature and set it to using the yours previously customized action.

And henceforward, you can reuse that action in this way in other scenes and projects.

Working with poses in the Action Editor



Pose markers in the Action Editor mode of the Dope Sheet editor.

Another destination to the Action Editor is a working with poses which are set in Pose mode during the rigging character armature.

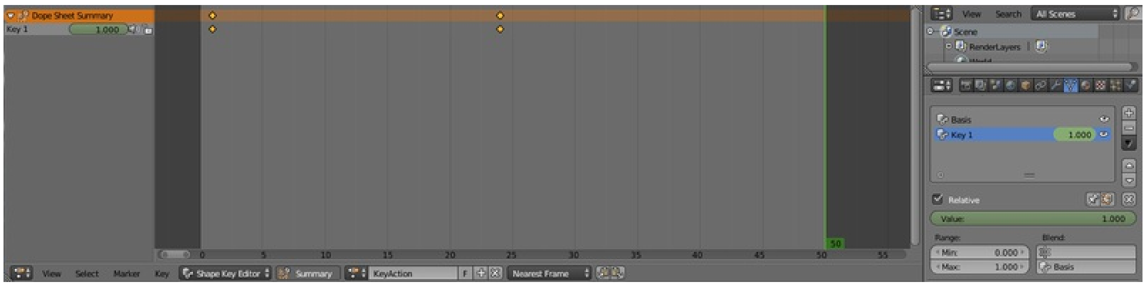
For these purposes *pose markers* are using in the Action Editor, which are detail described in section about animating armatures.

Shape Key Editor

Shape Key Editor mode of Dope Sheet Editor is using for working with shape key animation timing.

Shape keys are detail described here.

Interface



The Shape Key Editor mode of the Dope Sheet Editor window.

By their functionality, header, menus and window interface of the Action Editor is universal as for whole Dope Sheet Editor and about these you can see more details in corresponding sections:

- Dope Sheet Editor
- Graph Editor
- FCurve Editing
- Timeline
- Markers

One of the distinction in the Action Editor interface is the absence header elements and tools for channels.

Using Shape Key Editor

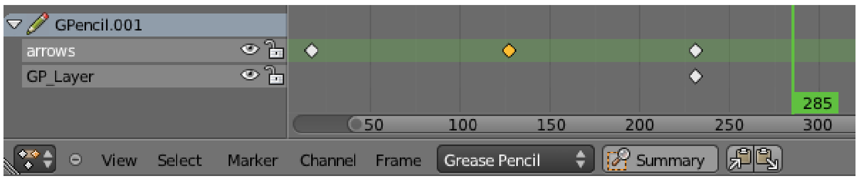
The Shape Key Editor mode of the Dope Sheet Editor is using similarly as the Action Editor mode with any animation actions.

Grease Pencil

Grease Pencil mode of Dope Sheet Editor is using for working with grease pencil drawings animation timing.

Grease Pencil is used to do basic pencil tests. See more info about this here.

Grease-Pencil block is loaded up in the DopeSheet editor for editing of the timings of the drawings. This is especially useful for animators blocking out shots, where the ability to re-time blocking is one of the main purposes of the whole exercise.



The Grease Pencil mode of the Dope Sheet Editor window

In Grease Pencil mode the DopeSheet editor displays on right its side a few “channels” with some “keyframes” on them. These “channels” are the layers, and the “keyframes” are the frames at which the layer has a sketch defined. They can be manipulated like any other action data in the DopeSheet.

Interface

By their functionality, header, menus and window interface of the Grease Pencil mode is universal as for whole Dope Sheet Editor and about these you can see more details in corresponding sections:

- Dope Sheet Editor
- Timeline
- Markers

One of the distinction in the Grease Pencil interface — the default for animation editors menu Key is cliped to menu Frame with only the duplicate and transform options.

Using Grease Pencil mode of Dope Sheet Editor

The Grease Pencil animation workflow is detail described here.

Mask

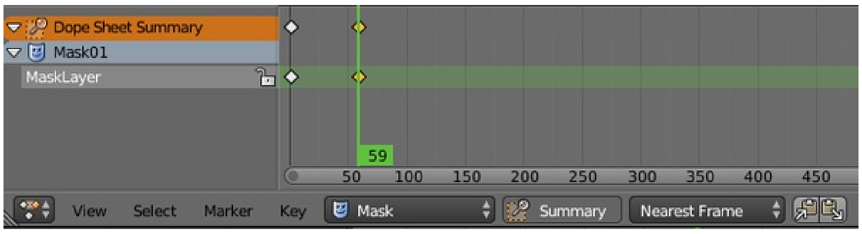
Mask is a grayscale raster image which is created from vector image. Artists interacts with vector image to create mask with needed shape and then it automatically gets rasterized.

Masks have many purposes. They can be used in a motion tracking workflow to mask out, or influence a particular object in the footage. They can be used for manual rotoscoping to pull a particular object out of the footage, or as a rough matte for green screen keying. Masks are independent from a particular image of movie clip, and so they can just as well be used for creating motion graphics or other effects in the compositor.

See more details about masks here

Masks can be driven over the time so that they follow some object from the footage, e.g. a running actor. For working with animation timing of masks in Blender there is a special mode in Dope Sheet Editor — Mask mode.

Interface



The Mask mode of the Dope Sheet Editor window.

By functionality, interface of the Mask is same as for whole Dope Sheet Editor and about this you can see more details in corresponding sections:

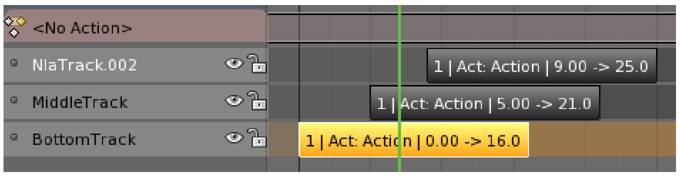
- Dope Sheet Editor
- Graph Editor
- FCurve Editing
- Timeline
- Markers

Non-Linear Animation Editor

The NLA editor can manipulate and repurpose actions, without the tedium of keyframe handling. Its often used to make broad, significant changes to a scene's animation, with relative ease. It can also repurpose, and 'layer' actions, which make it easier to organize, and version-control your animation.

Tracks

Tracks are the layering system of the NLA. At its most basic level, it can help organize strips. But it also layers motion much like an image editor layers pixels - the bottom layer first, to the top, last.


<No Action>
NlaTrack.002
MiddleTrack
BottomTrack
1 | Act: Action | 9.00 -> 25.0
1 | Act: Action | 5.00 -> 21.0
1 | Act: Action | 0.00 -> 16.0

Strips

There's three kinds of strips - Action, Transition, and Meta. Actions contain the actual keyframe data, Transitions will perform calculations between Actions, and Meta will group strips together as a whole.

Creating Action Strips

Any action used by the NLA first must be turned into an Action strip. This is done so by clicking the next to the action listed in the NLA. Alternatively, you can go to

Menu: Add → Action

Action Strip.

Creating Transition Strips

Select two or more strips on the same track, and go to

Menu: Add → Transition

Transition Strip.

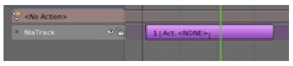
Grouping Strips into Meta Strips

If you find yourself moving a lot of strips together, you can group them into a Meta strip. A meta strip can be moved and duplicated like a normal strip.

Hotkey: ⇧ ShiftG

Menu: Add → Add Meta-Strips

Shift-select two or more strips..


Combine them into a meta strip.

A meta strip still contains the underlying strips. You can ungroup a Meta strip.

Hotkey: AltG

Menu: Add → Remove Meta-Strips

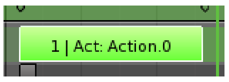
Editing Strips

The contents of Action strips can be edited, but you must be in 'Tweak Mode' to do so.

Hotkey: ⇆ Tab

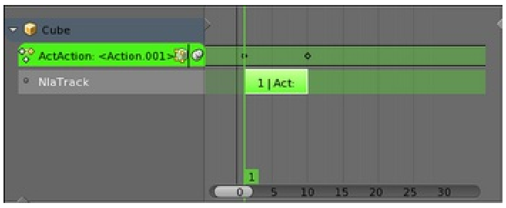
Menu: View → Enter Tweak Mode

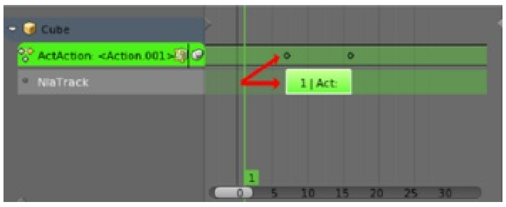
Strip in NLA mode..

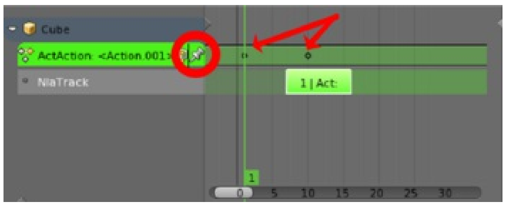


Strip in Tweak mode.

If you try moving the strip, while in edit mode, you'll notice that the keys will go along with it. On occasion, you'll prefer the keys to remain on their original frames, regardless of where the strip is. To do so, hit the 'unpin' icon, next to the strip.


Nla strip with pinned keys.


Strip moved, notice the keys move with it.


The unpinned keys return to their original frames.

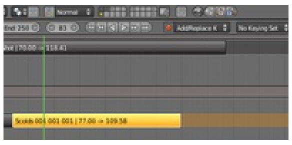
When your finished editing the strip, simply go to View > Exit Tweak Mode. Note the default key for this is Tab.

Re-Instancing Strips

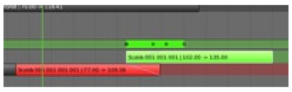
The contents' of one Action strip can be instanced multiple times. To instance another strip, select a strip, go to

Menu: Edit→ Duplicate Strips

Now, when any strip is tweaked, the others will change too. If a strip other than the original is tweaked, the original will turn to red.


Original strip.

Duplicated strip.


Duplicated strip being edited.

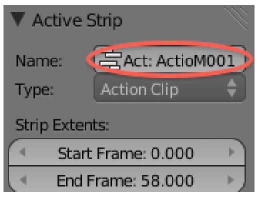
Strip Properties

Strip properties can be accessed via the NLA header.

Menu: View→ Properties

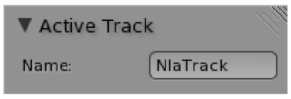
Renaming Strips

All strips can be renamed, in the "Active Track" section in the Strip Properties.



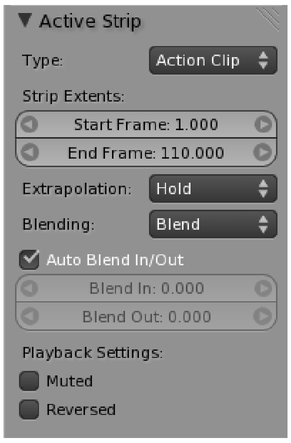
Active Track

This is which track the strip currently belongs to.



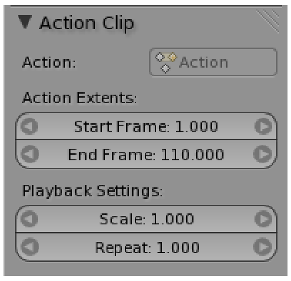
Active Strip

Elements of the strip itself. An Action Strip can be either an Action Clip, or a Transition Clip. Note that the 'Strip Extents' fields determine strictly the strip, and not the action. Also, the "Hold" value in the Extrapolation section means hold both beginning, and after. This can cause previous clips to not work, if checked.

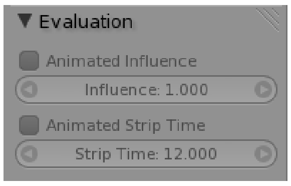

Active Strip
Type: Action Clip
Strip Extents:
Start Frame: 1.000
End Frame: 110.000
Extrapolation: Hold
Blending: Blend
Auto Blend In/Out [✓]
Blend In: 0.000
Blend Out: 0.000
Playback Settings:
Muted
Reversed

Active Action

This represents the 'object data' of the strip. Much like the transform values of an object.

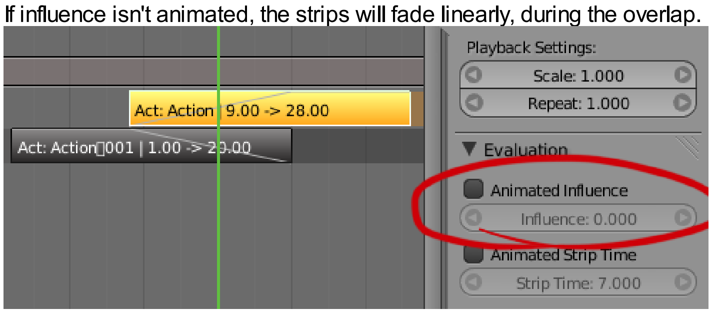

Action Clip
Action: Action
Action Extents:
Start Frame: 1.000
End Frame: 110.000
Playback Settings:
Scale: 1.000
Repeat: 1.000

Evaluation


Evaluation
Animated Influence
Influence: 1.000
Animated Strip Time
Strip Time: 12.000

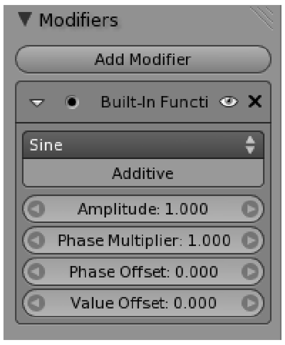
This determines the degree of influence the strip has, and over what time.

If influence isn't animated, the strips will fade linearly, during the overlap.


Act: Action | 9.00 -> 28.00
Act: Action.001 | 1.00 -> 20.00
Playback Settings:
Scale: 1.000
Repeat: 1.000
Evaluation
Animated Influence
Influence: 0.000
Animated Strip Time
Strip Time: 7.000

Strip Modifiers

Like its close cousins in mesh and graph editing, Modifiers can stack different combinations of effects for strips. Obviously there will be more to come on this.


Modifiers
Add Modifier
Built-In Functi
Sine
Additive
Amplitude: 1.000
Phase Multiplier: 1.000
Phase Offset: 0.000
Value Offset: 0.000

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Using Constraints in Animation

Constraints are a way to control an object’s properties (its location/rotation/scale), using either plain static values (like the “limit” ones), or (an)other object(s), called “targets” (like e.g. the “copy” ones).

Even though these constraints might be useful in static projects, their main usage is obviously in animation. There are two different aspects in constraints’ animation:

- You can control an object’s animation through the targets used by its constraints (this is a form of indirect animation).
- You can animate constraints’ settings

Controlling Animation with Constraints

This applies only to constraints using target(s). Indeed, these targets can then control the constraint’s owner’s properties, and hence, animating the targets will indirectly animate the owner.

This indirect “constraint” animation can be very simple, like for example with the Copy Location constraint, where the owner object will simply copy the location of its target (with an optional constant offset). But you can also have very complex behaviors, like when using the Action constraint, which is a sort of Animation Driver for actions!

We should also mention the classical Child Of constraint, which creates parent/child relationship. These relationships indeed imply indirect animation (as transforming the parent affects by default all its children). But the Child Of constraint is also very important, as it allows you to parent your objects to bones, and hence use Armatures to animate them!

Back to our simple Copy Location example, you can have two different behaviors of this constraint:

- When its Offset button is disabled (the default), the location of the owner is “absolutely” controlled by the constraint’s target, which means nothing (except other constraints below in the stack…) will be able to control the owner’s position. Not even the object’s animation curves.
- However, when the Offset button is enabled, the location of the owner is “relatively” controlled by the constraint’s target. This means that location’s properties of the owner are offset from the location of the target. And these owner’s location properties can be controlled e.g. by its Loc… curves (or actions, or NLA…)!

Example

Let’s use the Copy Location constraint and its Offset button. For example, you can make your owner (let’s call it `moon`) describe perfect circles centered on the `(0.0, 0.0, 0.0)` point (using e.g. pydriven LocX/LocY animation curves, see this page), and then make it copy the location of a target (called, I don’t know… `earth`, for example) – with the Offset button enabled. Congratulation, you just modeled a satellite in a (simplified) orbit around its planet… Just do the same thing with its planet around its star (which you might call `sun`, what do you think?), and why not, for the star around its galaxy…

Here is a small animation of a “solar” system created using (among a few others) the technique described above:

[video link]

Note that the this “solar” system is not realistic at all (wrong scale, the “earth” is rotating in the wrong direction around the “sun”, …).

You can download the the .blend file (File:ManAnimationTechsUsingConstraintsExSolarSys.blend) used to create this animation.

Animating Constraints Influence

More “classically”, you can also animate a few properties of each constraint using animation curves.

You only have two animation curves (see also this page):

- You can animate the Influence of a constraint. For example, in the “solar system” example above, I used it to first stick the camera to the “moon”, then to the “earth”, and finally to nothing, using two Copy Location constraints with Offset set, and their Influence cross-fading together…
- More anecdotal, you can also, for some constraints using an armature’s bone as target, animate where along this bone (between root and tip) lays the real target point (**0.0** means “full-root”, and **1.0**, “full-tip”).

Moving Objects on a Path

To make objects move along a path is a very common animation need. Think of a complex camera traveling, a train on his rails – and most other vehicles can also use “invisible” tracks! –, the links of a bicycle chain, etc. All these movements could obviously be done with standard Ipo curves, but this would be a nightmare! It’s much more easy and intuitive to define a path materializing the desired movement, and make your object(s) follow it.

Blender features you two different constraints to make an object follow a path, which have different ways to determine/animate the position of their owner along their path.

In Blender, any curve object can become a path. A curve becomes a path when its Path Animation button is enabled in the Curve data panel, but you don’t even have to bother about this: once a curve is selected as target for a “path” constraint, it automatically is enabled.

You can also directly add a “path” from the Add » Curve » Path menu entry (in a 3D view). This will insert in your scene a *three-dimensional* NURBS curve. This is an important point: by default, Blender’s curve are 2 dimensional, i.e. are laid on a plane, which is often not the desired behavior of a path. To turn a standard curve three-dimensional, enable its 3D button, in the same Curve and Surface editing panel.

One last curve property that is important for a path is its *direction*, which is, for three-dimensional ones, materialized by its small arrows. You can switch it with the Curve » Segments » Switch Direction menu entry (or W2 NumPad).

For more on editing path/curves, see the modeling chapter.

{{Note|Shapes on Curves|If you would rather like to have your object’s *shape* follow a path (like e.g. a sheet of paper inside a printer), you should use the Curve Modifier

Parenting Method

Older versions of Blender did not have constraints to make an object follow a path. They used a different method (deprecated, but still available), based on parenting.

To use this method, select the object that will follow the path, then ⇧ Shift select the curve, and use CtrlP to bring up the parenting menu. Choose Follow Path. The object will now be animated along the path.

The settings for the path animation are in the Path Animation panel of the Curve properties panel.

Frames
: Defines the number of frames it takes for the object to travel the path.

Evaluation Time
: Defines current frame of the animation. By default it is linked to the global frame number, but could be keyframed to give more control over the path animation.

Follow
: Causes the curve path children to rotate along the curvature of the path.

Radius
: Causes the curve path child to be scaled by the set curve radius. See Curve Extruding

Offset Children
: Causes the animation to be offset by the curve path child's time offset value, which can be found in its Animation Hacks section of the Object Panel.

The Follow Path Constraint

The Follow Path constraint implements the most “classical” technique. By default, the owner object will walk the whole path only once, starting at frame one, and over **100** frames. You can set a different starting frame in the Offset field of the constraint panel, and change the length (in frames) of the path using its Frames property (Curve and Surface panel).

But you can have a much more precise control over your object’s movement along its path by keyframing or defining a Speed animation curve for the path's Evaluation Time attribute. This curve maps the current frame to a position along the path, from **0.0** (start point) to **1.0** (end point).

For more details and examples, see the Follow Path constraint page.

The Clamp To Constraint

Another method of keeping objects on a path is to use the Clamp To constraint, which implements a more advanced technique. To determine where along the path should lay its owner, its uses the *location of this owner* along a given axis. So to animate the movement of your owner along its target path, you have to animate some way (Ipo curves or other indirect animation) its location.

This implies that here, the length of the path have no more any effect – and that by default, the object is static somewhere on the path!

For more details and examples, see the Clamp To constraint page.

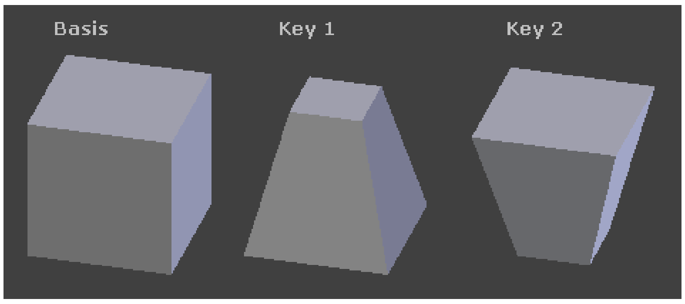
Page status (reviewing guidelines)

Void page
Proposed fixes: none

Shape Keys

Introduction

Shape Keys are used on Objects like Mesh, Curve, Surface, Lattice.
They are used to deform the object vertices into a new shape.



A mesh with different shape keys applied.

There are two types of Shape Keys.

Relative
: Which are relative to the Basis or selected shape key.
They are mainly used as, for limb joints, muscles, or Facial Animation.

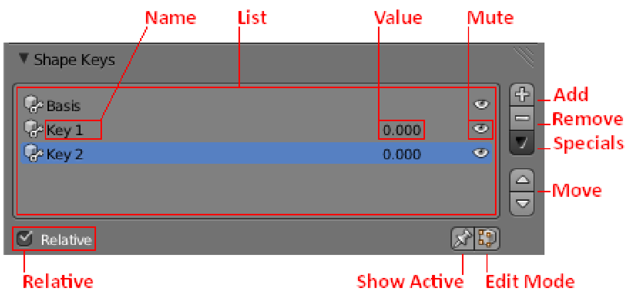
Absolute
: Which are relative to the previous and next shape key.
They are mainly used to deform the objects into different shapes over time.

The shape key data, the deformation of the objects vertices, is usually modified in the 3D View by selecting a shape key, then moving the object vertices to a new position.

Shape Keys Panel

Mode: All modes

Panel: Properties, Object Data, Shape Keys



Shape Keys. Options.

Relative
: Set the shape keys to Relative or Absolute.

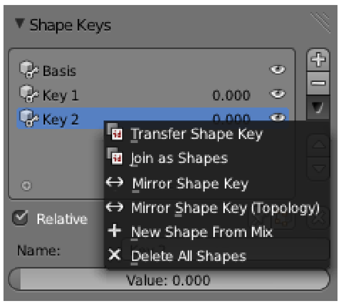
Name
: Name of the Shape Key.

Value
: Current Value of the Shape Key (0.0 to 1.0).

Mute
: This visually disables the shape key in the 3D view.

Add
: Add a new shape key to the list.

Remove
: Remove a shape key from the list.



Shape Keys Specials.

Specials
A menu with some operators.

Transfer Shape Key
Transfer the active 'Shape Key' from a different object.
Select two objects, the active Shape Key is copied to the active object.

Join as Shapes
Transfer the 'Current Shape' from a different object.
Select two objects, the Shape is copied to the active object.

Mirror Shape Key
If your mesh is nice and symmetrical, in Object Mode, you can mirror the shape keys on the X axis.
This wont work unless the mesh vertices are perfectly symmetrical.
Use the Mesh » Symmetrize function in Edit Mode.

Mirror Shape Key (Topology)
This is the same as Mirror Shape Key though it detects the mirrored vertices based on the topology of the mesh.
The mesh vertices dont have to be perfectly symmetrical for this one to work.

New Shape From Mix
Add a new shape key with the current deformed shape of the object.

Delete All Shapes
Delete all shape keys.

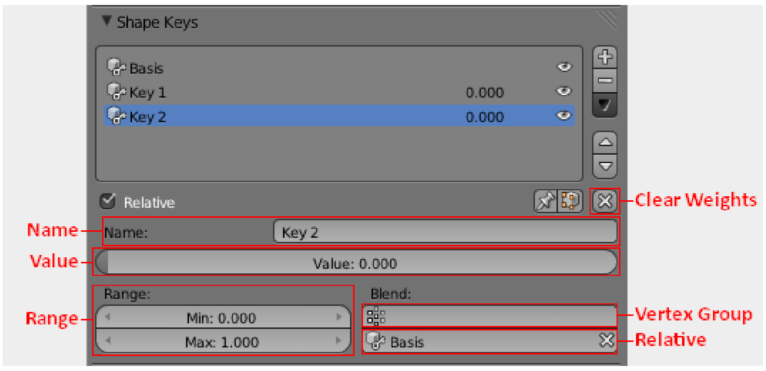
Move
Move shape key up or down in the list.

Show Active
Show the shape of the active shape key in the 3D View.
ShowActive is enabled while the object is in *Edit Mode*, unless the setting below is enabled.

Edit Mode
Modify the shape key settings while the object is in *Edit mode*.

Relative Shape Keys

Relative shape keys deform from a selected shape key.
By default all relative shape keys deform from the first shape key called the Basis shape key.



Relative Shape Keys. Options.

Clear Weights
Set all values to 0.

Name
Name of the active shape key.

Value
Value of the active shape key.

Range

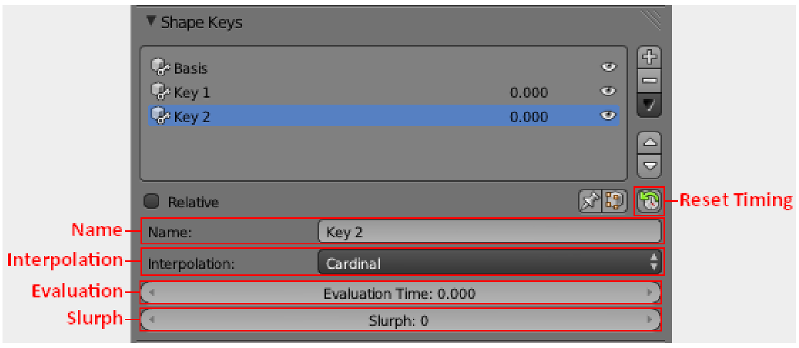
Min and Max range of the active shape key value.

Vertex Group
: Limit the active shape key deformation to a vertex group.

Relative
: Select the shape key to deform from.

Absolute Shape Keys

Absolute shape keys deform from the previous and to the next shape key.
They are mainly used to deform the object into different shapes over time.



Name: Key 2
Interpolation: Cardinal
Evaluation Time: 0.000
Slurph: 0

Absolute Shape Keys. Options.

Reset Timing
: Reset the timing for absolute shape keys.
For example, if you have the shape keys, Basis, Key_1, Key_2, in that order.
Reset Timing will loop the shapekeys, and set the shape key frames to +0.1.

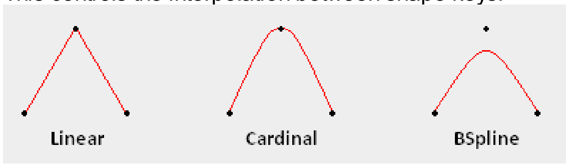
Basis 0.1
Key_1 0.2
Key_2 0.3

Evaluation Time will show this as frame*100.

Basis 10.0
Key_1 20.0
Key_2 30.0

Name
: Name of the active shape key.

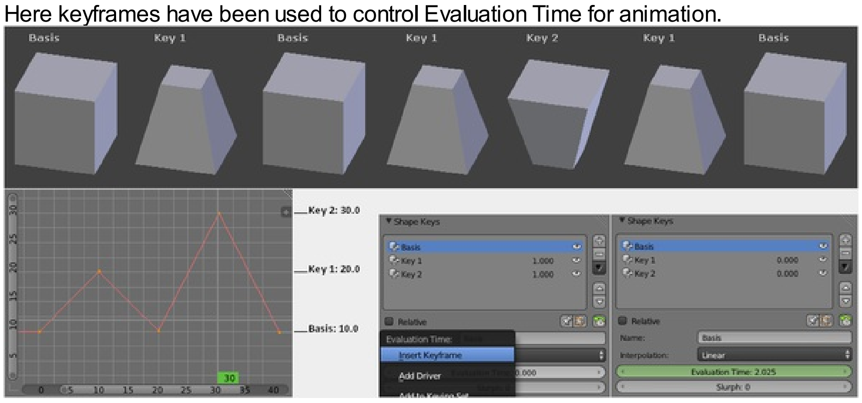
Interpolation
: This controls the interpolation between shape keys.



Evaluation Time
: This is used to control the shape key influence.
For example, if you have the shape keys, Basis, Key_1, Key_2, in that order,and you reset timing.

Basis 10.0
Key_1 20.0
Key_2 30.0

You can control the shape key influence with Evaluation Time.
Here keyframes have been used to control Evaluation Time for animation.



Slurph
: Quote 2.66 "Create a delay (in frames) in applying key positions, first vertex goes first."
As far as i can tell this doesnt anything in 2.66. Im not sure what it used to do in 2.4x.
2.4 Slurph

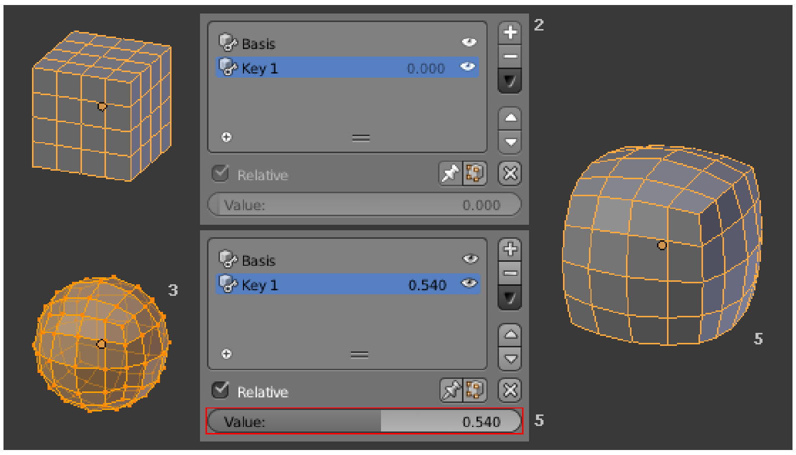
Workflow For Relative Shape Keys

This example shows you how to make a cube mesh transform in to a sphere.

1. In *Object Mode* add two shape keys via the *Shape Key Panel*.

   Basis is the rest shape. *Key 1* will be the new shape.

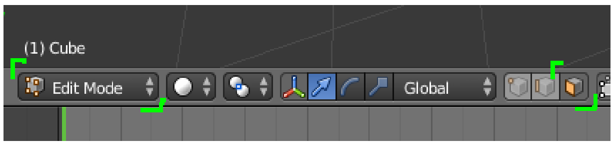
2. With *Key 1* selected, switch to *Edit Mode*.
3. Press ⇧ ShiftAltS *To Sphere*, move the mouse right, then LMB.
4. Switch to *Object Mode*.
5. Set the *Value* for *Key 1* to see the transformation between the shape keys.

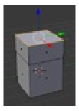


Shape Key workflow.

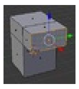
Workflow For Absolute Shape Keys

- Select the default Cube.
- Switch to Edit Mode.
- Switch to Face Select mode (if you are not already in it)



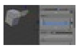


- Select the top face.
- Extrude up E 1 LMB.

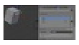


- Select a side face on the top half. (the one at x=1 if possible)
- Extrude out E 1 LMB.
- Switch back to Object Mode.

- Add a basis shape keys and two more via the + button on the Shape Key Panel.
- Uncheck the Relative checkbox.
- Click the Reset Timing button.
- Switch to Edit Mode.



- Select shape key Key 2 to edit the third shape key.
- Select the extruded side face and G Z 1 LMB



- Select shape key Basis to edit the first shape key.
- Select the extruded size face and S 0.5 LMB, then G X -1 LMB.

- Switch to Object Mode.
- Drag the Evaluation Time slider to make its value vary from 10 to 30.

Doc Absolute SK Workflow 7.GIF

More Details On Absolute Shape Keys

The thing to remember about absolute shape keys is that they are incomplete until you click the Reset Timing button. When you create a shape key its "frame" property is zero (https://developer.blender.org/T39897), which is a completely useless value. This frame value is not displayed on the UI so you can't easily tell if something is wrong or screwy until your animation starts misbehaving.

The number displayed to the right of the key name is the value and is used in relative shape keys. It has no effect on absolute shape keys, so ignore it.

When you reset the timings blender iterates through the shape keys assigning them frame values incrementing by 0.1 from key to key.

name	frame	evaluation time
Basis	0.1	10
Key 1	0.2	20
Key 2	0.3	30
Key 3	0.4	40

If you delete a shape key this does not automatically alter the frame values assigned to remaining shape keys.

name	frame	evaluation time
Basis	0.1	10
Key 1	0.2	20
Key 3	0.4	40

The Evaluation Time is how you choose which shape key is active, and how active it is. The interesting values range from 10 .. (n*10) where n is the number of shape keys. (assuming you have not deleted or added any keys since the last Reset Timing). If you are using shape keys for animation, 99% of the time you will be putting keyframes on this Evaluation Time field.

Remember: if you are having problems with your absolute shape keys, there is a good chance that you need to Reset Timing.

Shape Key Operators

3D View > Edit Mode > Header > Mesh > Vertices > Shape Propagate
	Apply selected vertex locations to all other shape keys.

3D View > Edit Mode > Header > Mesh > Vertices > Blend From Shape
    Blend in shape from a shape key.

See Also

- 2.4 Shape Keys
- 2.4 Editing Shape Keys
- 2.4 Animating Shape Keys
- 2.4 Shape Keys Examples
- Addon: Corrective Shape Key

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Motion Capture

On this moment see more details here.

Animating Lamps Properties

As of Blender 2.5, Everything is animatable. Read more about keyframes Here.

Example

Let’s illustrate this with a flying torch deep in a cave.

We won’t detail the cave and torch creation – the first one is an deformed icosphere with Subsurf and Displace modifiers, and the second one, a cylinder scaled and subdivided several times in its length, with a particle system to materialize its fire.

The torch will be the only light source of the scene. Add four Lamp lamps, all using the same lamp datablock. Place them around the tip of the torch, and parent them to it. Give them an orange color (e.g. `(1.0, 0.8, 0.4)`), a short Distance (**2.0**, but this depends on the size of your cave!), and an Inverse Square falloff. Also let them cast ray shadows (soft shadows, if you have enough computing power…).

Right click on the Energy parameter and Insert Keyframe to create an Fcurve, then open the graph editor to edit the keyframes. You can for example start at zero (no energy, the scene is whole black), give a short and intense flash (**10** over a few frames) to simulate a sort of lightning lighting the fire, then back to very low (**0.25**), and then a gently varying curve over the rest of the scene, to simulate the irregularities of the flames.

You might also use the same animation to control the particle system emmission, to synchronize the quantity of particles with the luminosity of the lamps.

Once your torch is flying, you should get something as shown below – you can download the blend file File:ManAnimationTechsLampExFlyingTorch.blend.

[video link]

Animating Cameras

As of Blender 2.5, Everything is animatable. Read more about keyframes Here.

Example

As an example, we are going to create a nice and impressive camera effect, which you can see e.g. in the first part of the Lord of the Ring: the *transtrav*. Basically, the idea is to combine a forward zoom with a backward traveling (or conversely), both controlled such as the point of interest keeps its scale in the image, while its environment scales up or down, depending whether it is nearer or more far from the camera…

Create a scene with a ground, and some objects laying on it.

Add a camera, place it as you like for the beginning of the transtrav (your “key” object should be more or less at the center of the picture, it’s easier to handle!). As we are going to do a “forward” transtrav, you should use a quite long lens at start. Go to frame **10**, and insert a keyframe for both the location (and optionally the rotation) of the camera and its focal length.

Now, go to frame **140**, and move forward your camera to your key object. Insert another keyframe for its position, and adjust its focal length until your key object have the same visual dimensions as at the beginning. Add a key to both attributes.

This won’t give you a fully perfect transtrav – to get such one, you would have to dive into trigonometric maths… But the result is visually quite satisfying.

[video link]

Animating Material Attributes

As of Blender 2.5, Everything is animatable. Read more about keyframes Here.

Before reading this page, you should know about Blender's materials – if not, read this chapter first!

Animated materials can be a very powerful tool, for many different purposes. For example, you can use them to simulate the color changes of a chameleon's skin, a video screen lighting up, the surface of a river or lake, a light lighting up (with an halo material), etc., etc.

The possibilities are nearly unlimited. For example, you could keyframe the Glossiness of raytraced reflection/transparency, which would enable the simulation of condensation spreading over a mirror or window…).

Example

As an illustration, we'll create a simple "psychedelic" background. This obviously won't demonstrate all possibilities of material animation – but I think this would need at least a whole book!

Add a plane and a camera, such that the plane faces the camera and covers the whole view.

Add a material to the plane. As we won't use any light, set its Emit value to **1.0**.

Create Fcurves for R, G and B, with a few random control points, all in the `[0.0, 1.0]` range. Manage to have three different length between the first and last keyframes, and enable the Cyclic extend mode (E2 NumPad). This way, with the three curves cycling over various periods, you'll get a never-the-same color animation! Unless you want to get a "time-tileable" animation, in which case you should manage to get exactly the same color at start and end… You can also create an Emit Fcurve, e.g. to create a fade in/out…

Now, let's add a bit of fun in this plain colored background. Add a texture to the material and, in the Texture sub-context, select a procedural texture (DistortedNoise, for example, but any one will work – follow your taste!), and set it to your liking.

Back in the Material sub-context, choose to what you want to map the texture – for this example, I chose to map it to diffuse color in Difference mode, in a first texture channel, and to emit value in Multiply mode, in a second texture channel.

Finally, animate the Z offset of the mapping of both channels (define a first OfsZ Fcurve, and use the copy/paste buttons to exactly copy it to the second texture channel's curve). Here again, you can either have two different values for start and end, or the same if you want a cyclic animation…

Usually, you will create a slow, linear variation of the Z offset (i.e. a straight curve with low gradient), e.g. a decay of **1.0** over **500** to **1000** frames, but the only way to find the good value is to make preview renders!

You should get something similar to what shown below. You can download the blend file

File:ManAnimationTechsMaterialExPshychedelic.blend.

[video link]

Introduction to Physics Simulation

This chapter covers various advanced Blender effects, often used to simulate real physical phenomena, such as:

- Smoke
- Rain
- Dust
- Cloth
- Water
- Jello

Particle Systems can be used to simulate many things: hair, grass, smoke, flocks.

Hair is a subset of the particle system, and can be used for strand-like objects, such as hair, fur, grass, quills, etc.

Soft Bodies are useful for everything that tends to bend, deform, in reaction to forces like gravity or wind, or when colliding with other objects… It can be used for skin, rubber, and even clothes, even though there is separate Cloth Simulation specific for cloth-like objects.

Rigid Bodies can simulate dynamic objects that are fairly rigid.

Fluids, which include liquids and gasses, can be simulated, including Smoke.

Force Fields can modify the behavior of simulations.

Gravity

Gravity is a global setting that is applied the same to all physics systems in a scene, which can be found in the scene tab. This value is generally fine left at its default value, at -9.810 in the Z axis, which is the force of gravity in the real world. Lowering this value would simulate a lower or higher force of gravity. Gravity denoted g, measurement [m×s^{-2}]).

Gravity is practically same around whole **Earth**. South and North poles have g = 9.832 m×s^{-2}, on the equator g = 9.780 m×s^{-2}. For detail computing of falling in Prague or Boston use 9.81373. For rendering scenes from **Moon** use value 6 times smaller, e.g. 1.622 m×s^{-2}. The most popular and probably not colonized **Mars** has g = 3.69.

Note that you can scale down the gravity value per physics system in the Field Weights tab.

Force Fields

Force Fields offer a way to add extra movement to dynamic systems. Particles, Soft Bodies, Rigid Bodies and Cloth objects can all be affected by forces fields. Force Fields automatically affect everything. To remove a simulation or particle system from their influence, simply turn down the influence of that type of Force Field in its Field Weights panel.

- All types of objects and particles can generate fields, but only curve object can bear Curve Guides fields.
- Force Fields can also be generated from particles. See Particle Physics
- The objects need to share at least one common layer to have effect.

You may limit the effect on particles to a group of objects (see the Particle Physics page).

Creating a Force Field

Mode: Object Mode

Panel: Object context → Physics sub-context → Fields

Hotkey: F7

To create a single Force Field, you can select Add » Force Field and select the desired force field. This method creates an Empty with the force field attached.

To create a field from an existing object you have to select the object and change to the Physics sub-context. Select the field type in the Fields menu.

The fields have many options in common, these common options are explained for the Spherical field.

Note

After changing the fields (Fields panel) or deflection (Collision panel) settings, you have to recalculate the particle, softbody or cloth system (Free Cache), this is not done automatically. You can clear the cache for all selected objects with CtrlB → Free cache selected.

Particles react to all kind of Force Fields, Soft Bodies only to Spherical/Wind/Vortex (they react on Harmonic fields but not in a useful way).

Common Field Settings

Most Fields have the same settings, even though they act very differently. Settings unique to a field type are described below. Curve Guide and Texture Fields have very different options.

Shape
: The field is either a Point, with omnidirectional influence, or a Plane, constant in the XY-plane, changes only in Z direction.

Strength
: The strength of the field effect. This can be positive or negative to change the direction that the force operates in. A force field’s strength is scaled with the force object’s scale, allowing you to scale up and down scene, keeping the same effects.

Flow
: Convert effector force into air flow velocity.

Noise
: Adds noise to the strength of the force.

Seed
: Changes the seed of the random noise.

Effect Point
: You can toggle the field's effect on particle Location and Rotation

Collision Absorption
: Force gets absorbed by collision objects.

Falloff

Here you can specify the shape of the force field (if the Fall-off Power is greater than 0).

Sphere
: Falloff is uniform in all directions, as in a sphere.

Tube
: Fall off results in a tube shaped force field.
: The Field's Radial falloff can be adjusted, as well as the Minimum and Maximum distances of the field.

Cone
: Fall off results in a cone shaped force field. Additional options are the same as those of Tube options.

Z Direction
: Fall-off can be set to apply only in the direction of the positive Z Axis, negative Z Axis, or both.

Power (Power)

How the power of the force field changes with the distance from the force field. If `r` is the distance from the center of the object, the force changes with $1/r^{Power}$. A Fall-off of 2 changes the force field with $1/r^2$, which is the falloff of gravitational pull.

Max Distance
: Makes the force field only take effect within a specified maximum radius (shown by an additional circle around the object).

Min Distance
: The distance from the object center, up to where the force field is effective with full strength. If you have a Fall-off of 0 this parameter does nothing, because the field is effective with full strength up to Max Dist (or the infinity). Shown by an additional circle around the object.

Field Types

Force

The Force field is the simplest of the fields. It gives a constant force towards (positive strength) or away from (negative strength) the object's center. Newtonian particles are attracted to a field with negative strength, and are blown away from a field with positive strength.

For Boids a field with positive strength can be used as a *Goal*, a field with negative strength can be used as *Predator*. Whether Boids seek or fly goals/predators depends on the Physics settings of the Boids.

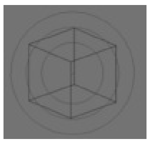


Image 2b: Spherical field indicator.

Wind

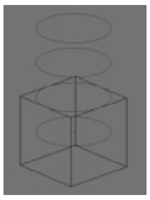


Image 3a: Wind field indicator.

Wind gives a constant force in a single direction, along the force object's local Z axis. The strength of the force is visualized by the spacing of the circles shown.

Vortex Field

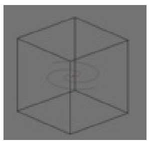


Image 3b: Vortex field indicator.

Vortex fields give a spiraling force that twists the direction of points around the force object's local Z axis. This can be useful for making a swirling sink, or tornado, or kinks in particle hair.

Magnetic

This field depends on the speed of the particles. It simulates the force of magnetism on magnetized objects.

Harmonic

The source of the force field is the zero point of a harmonic oscillator (spring, pendulum). If you set the Damping parameter to 1, the movement is stopped in the moment the object is reached. This force field is really special if you assign it to particles.

Rest Length
: Controls the rest length of the harmonic force.

Multiple Springs
: Causes every point to be affected by multiple springs.

Normally every particle of the field system influences every particle of the target system. Not with Harmonic! Here every target particle is assigned to a field particle. So particles will move to the place of other particles, thus forming shapes. Tutorial: Particles forming Shapes.

Charge

It is similar to spherical field except it changes behavior (attract/repulse) based on the effected particles charge field (negative/positive), like real particles with a charge. This mean this field has only effect on particles that have also a Charge field (else, they have no “charge”, and hence are unaffected)!

Lennard-Jones

This field is a very short range force with a behavior determined by the sizes of the effector and effected particle. At a distance smaller than the combined sizes the field is very repulsive and after that distance it’s attractive. It tries to keep the particles at an equilibrium distance from each other. Particles need to be at a close proximity to each other to be effected by this field at all.

Particles can have for example both a charge and a Lennard-Jones potential - which is probably something for the nuclear physicists amongst us.

Texture field

You can use a texture force field to create an arbitrarily complicated force field, which force in the 3 directions is color coded. Red is coding for the x-axis, green for the y-axis and blue for the z-axis (like the color of the coordinate axes in the 3D window). A value of 0.5 means no force, a value larger than 0.5 acceleration in negative axis direction (like -Z), a value smaller than 0.5 acceleration in positive axis direction (like +Z).

Texture mode
: This sets the way a force vector is derived from the texture.

  RGB
  : Uses the color components directly as the force vector components in the color encoded directions. You need an RGB texture for this, e.g. an image or a colorband. So a Blend texture without a colorband would not suffice.

  Gradient
  : Calculates the force vector as the 3d-gradient of the intensity (grayscale) of the texture. The gradient vector always points to the direction of increasing brightness.

  Curl
  : Calculates the force vector from the curl of the 3d-rgb texture (rotation of rgb vectors). This also works only with a color texture. It can be used for example to create a nice looking turbulence force with a color clouds texture with perlin noise.

Nabla
: It is the offset used to calculate the partial derivatives needed for Gradient and Curl texture modes.

Use Object Coordinates
: Uses the emitter object coordinates (and rotation & scale) as the texture space the particles use. Allows for moving force fields, that have their coordinates bound to the location coordinates of an object.

Root Texture Coordinates
: This is useful for hair as it uses the texture force calculated for the particle root position for all parts of the hair strand.

2D
: The 2D button disregards the particles z-coordinate and only uses particles x&y as the texture coordinates.

Remember that only procedural texture are truly 3D.

Examples

- A single colored texture 0.5/0.0/0.5 creates a force in the direction of the positive y-axis, e.g. hair is orientated to the y-axis.
- A blend texture with colorband can be used to created a force “plane”. E.g. on the left side 0.5/0.5/0.5, on the right side 1.0/0.5/0.5 you have a force plane perpendicular to XY (i.e. parallel to Z). If you use an object for the coordinates, you can use the object to push particles around.
- An animated wood texture can be used to create a wave like motion.

Curve Guide

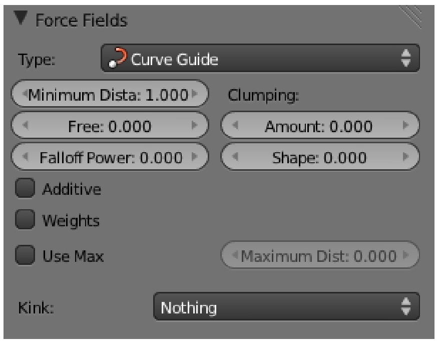

Force Fields
Type: Curve Guide
Minimum Dista: 1.000
Clumping:
Free: 0.000
Amount: 0.000
Falloff Power: 0.000
Shape: 0.000
Additive
Weights
Use Max
Maximum Dist: 0.000
Kink: Nothing

Image 4a: A Curve Guide field.

Curve objects can be the source of a Curve Guide field. You can guide particles along a certain path, they don't affect Softbodys. A typical scenario would be to move a red blood cell inside a vein, or to animate the particle flow in a motor. You can use Curve Guides also to shape certain hair strands - though this may no longer be used as often now because we have the Particle Mode. Since you can animate curves as Softbody or any other usual way, you may build very complex animations while keeping great control and keeping the simulation time to a minimum.

The option Curve Follow does not work for particles. Instead you have to set Angular Velocity (in the Physics panel of the Particle sub-context) to Spin and leave the rotation constant (i.e. don't turn on Dynamic).

Curve Guides affect all particles on the same layer, independently from their distance to the curve. If you have several guides in a layer, their fields add up to each other (the way you may have learned it in your physics course). But you can limit their influence radius:

Minimum Distance
: The distance from the curve, up to where the force field is effective with full strength. If you have a Fall-off of 0 this parameter does nothing, because the field is effective with full strength up to MaxDist (or the infinity). MinDist is shown with a circle at the endpoints of the curve in the 3D window.

Free
: Fraction of particle life time, that is not used for the curve.

Fall-off
: This setting governs the strength of the guide between MinDist and MaxDist. A Fall-off of 1 means a linear progression.

A particle follows a Curve Guide during it's lifetime, the velocity depends from it's lifetime and the length of the path.

Additive
: If you use Additive, the speed of the particles is also evaluated depending on the Fall-off.

Weights
: Use Curve weights to influence the particle influence along the curve.

Maximum Distance/Use Max
: The maximum influence radius. Shown by an additional circle around the curve object.

The other settings govern the form of the force field along the curve.

Clumping Amount
: The particles come together at the end of the curve (1) or they drift apart (-1).

Shape
: Defines the form in which the particles come together. +0.99: the particles meet at the end of the curve. 0: linear progression along the curve. -0.99: the particles meet at the beginning of the curve.

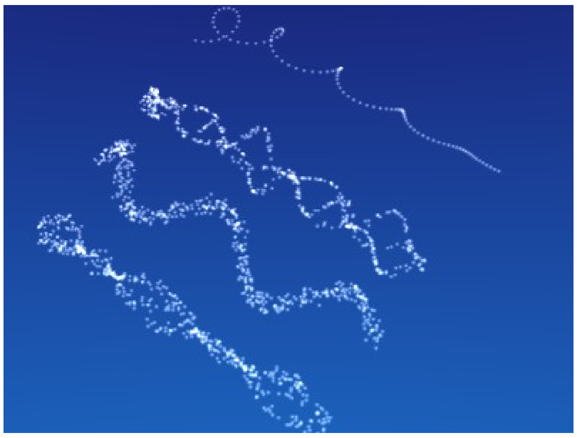


Image 4b: Kink options of a curve guide. From left to right: Radial, Wave, Braid, Roll.
Animation

With the drop down box Kink, you can vary the form of the force field:

Curl
: The radius of the influence depends on the distance of the curve to the emitter.

Radial
: A three dimensional, standing wave.

Wave
: A two dimensional, standing wave.

Braid
: Braid.

Roll
: A one dimensional, standing wave.

It is not so easy to describe the resulting shapes, I hope it's shown clearly enough in (*Image 4b*).

Frequency
: The frequency of the offset.

Shape

Adjust the offset to the beginning/end.

Amplitude
: The Amplitude of the offset.

Boid

Boid probably comes from theoretical works. Boids is an artificial life program, developed by Craig Reynolds in 1986, which simulates the flocking behaviour of birds. His paper on this topic was published in 1987 in the proceedings of the ACM SIGGRAPH conference. The name refers to a "bird-like object", but its pronunciation evokes that of "bird" in a stereotypical New York accent. As with most artificial life simulations, Boids is an example of emergent behavior; that is, the complexity of Boids arises from the interaction of individual agents (the boids, in this case) adhering to a set of simple rules. The rules applied in the simplest Boids world are as follows: separation: steer to avoid crowding local flockmates alignment: steer towards the average heading of local flockmates cohesion: steer to move toward the average position (center of mass) of local flockmates More complex rules can be added, such as obstacle avoidance and goal seeking.

Turbulence

Create a random turbulence effect with a 3d noise.

Size
: Indicates the scale of the noise.

Global
: Makes the size and strength of the noise relative to the world, instead of the object it is attached to.

Drag

Drag is a force that works to resist particle motion by slowing it down.

Linear
: Drag component proportional to velocity.

Quadratic
: Drag component proportional to the square of the velocity.

Links

- Wind & Deflector force update 2.48
- Particle options and guides (v2.40)

Collisions

Particles, Soft Bodies and Cloth objects may collide with mesh objects. Boids try to avoid Collision objects.

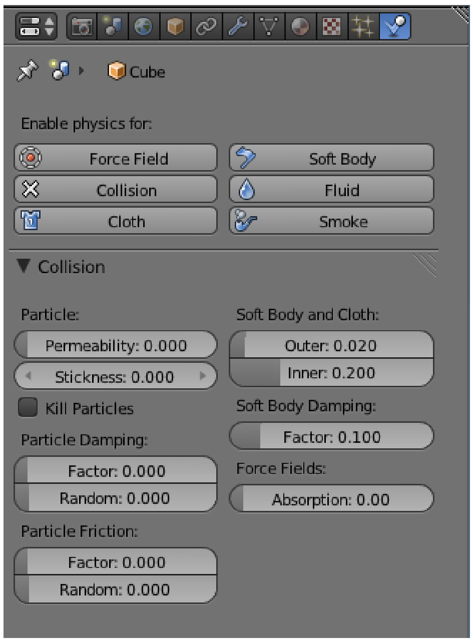
- The objects need to share at least one common layer to have effect.
- You may limit the effect on particles to a group of objects (in the Field Weights panel).
- *Deflection* for softbody objects is difficult, they often penetrate the colliding objects.
- Hair particles ignore deflecting objects (but you can animate them as softbodies which take deflection into account).

If you change the deflection settings for an object you have to recalculate the particle, softbody or cloth system (Free Cache), this is not done automatically. You can clear the cache for all selected objects with CtrlB → Free cache selected.

Mode: Object Mode

Panel: Object context → Physics sub-context → Collision

Options


Cube

Enable physics for:

Force Field | Soft Body
Collision | Fluid
Cloth | Smoke

Collision

Particle:
Permeability: 0.000
Stickness: 0.000
Kill Particles
Particle Damping:
Factor: 0.000
Random: 0.000
Particle Friction:
Factor: 0.000
Random: 0.000

Soft Body and Cloth:
Outer: 0.020
Inner: 0.200
Soft Body Damping:
Factor: 0.100
Force Fields:
Absorption: 0.00

Image 1: Collision panel in the Physics sub-context.

Permeability
: Fraction of particles passing through the mesh. Can be animated with Object Ipos, Perm channel.

Stickiness
: How much particles stick to the object.

Kill Particles
: Deletes Particles upon impact.

Damping Factor
: Damping during a collision (independent of the velocity of the particles).

Random damping
: Random variation of damping.

Friction Factor
: Friction during movements along the surface.

Random friction
: Random variation of friction.

Image 1b: A softbody vertex colliding with a plane.

Soft Body and Cloth Interaction

Outer
: Size of the outer collision zone.

Inner
: Size of the inner collision zone (padding distance). **Only effective for softbodies**, cloth only use outer collision value.

Outside and inside is defined by the face normal, depicted as blue arrow in (*Image 1b*).

Damping Factor
: Damping during a collision.

Softbody collisions are difficult to get perfect. If one of the objects move too fast, the soft body will penetrate the mesh. See also the section about Soft Bodies.

Force Field Interaction

Absorption
: A deflector can also deflect effectors. You can specify some collision/deflector objects which deflect a specific portion of the effector force using the Absorption value. 100% absorption results in no force getting through the collision/deflector object at all. If you have 3 collision object behind each other with e.g. 10%, 43% and 3%, the absorption ends up at around 50% (`100×(1-0.1)×(1-0.43)×(1-0.03)`).

Examples

Image 2: Deflected Particles.

Here is a Meta object, dupliverted to a particle system emitting downwards, and deflected by a mesh cube:

Hints

- Make sure that the normals of the mesh surface are facing towards the particles/points for correct deflection.
- Hair particles react directly to force fields, so if you use a force field with a short range you don’t need necessarily collision.
- Hair particles avoid their emitting mesh if you edit them in Particle mode. So you can at least model the hair with collision.

Particles

Particles are lots of items emitted from mesh objects, typically in the thousands. Each particle can be a point of light or a mesh, and be joined or dynamic. They may react to many different influences and forces, and have the notion of a lifespan. Dynamic particles can represent fire, smoke, mist, and other things such as dust or magic spells.

Hair type particles are a subset of regular particles. Hair systems form strands that can represent hair, fur, grass and bristles.

You see particles as a Particle modifier, but all settings are done in the Particle tab.

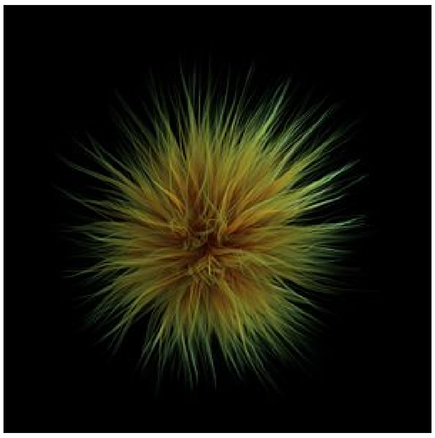


Image 1: Some fur made from particles (Blend file).

Particles generally flow out from their mesh into space. Their movement can be affected by many things, including:

- Initial velocity out from the mesh.
- Movement of the emitter (vertex, face or object) itself.
- Movement according to “gravity” or “air resistance”.
- Influence of force fields like wind, vortexes or guided along a curve.
- Interaction with other objects like collisions.
- Partially intelligent members of a flock (herd, school, …), that react to other members of their flock, while trying to reach a target or avoid predators.
- Smooth motion with softbody physics (only Hair particle systems).
- Or even manual transformation with Lattices.

Particles may be rendered as:

- Halos (for Flames, Smoke, Clouds).
- Meshes which in turn may be animated (e.g. fish, bees, …). In these cases, each particle “carries” another object.
- Strands (for Hair, Fur, Grass); the complete way of a particle will be shown as a strand. These strands can be manipulated in the 3D window (combing, adding, cutting, moving, etc).

Every object may carry many particle systems. Each particle system may contain up to 100.000 particles. Certain particle types (Hair and Keyed) may have up to 10.000 children for each particle (children move and emit more or less like their respective parents). The size of your memory and your patience are your practical boundaries.

Incompatibility with Prior Versions
There are many differences between the “old” particle system that was used up to and including version 2.45, and the “new” particle system. There are many things possible now that could not be done with the old system. The new system is incompatible to the old system, though Blender tries to convert old particle systems, which works only to some extent. The old system is most like the new Emitter system (keep reading to find out what that is). If you are using an old version of Blender 2.45 and previous, click here to access the old documentation.

Workflow

The process for working with standard particles is:

1. Create the mesh which will emit the particles.
2. Create one or more Particle Systems to emit from the mesh. Many times, multiple particle systems interact or merge with each other to achieve the overall desired effect.
3. Tailor each Particle System’s settings to achieve the desired effect.
4. Animate the base mesh and other particle meshes involved in the scene.
5. Define and shape the path and flow of the particles.
6. For Hair particle systems: Sculpt the emitter’s flow (cut the hair to length and comb it for example).
7. Make final render and do physics simulation(s), and tweak as needed.

Creating a Particle System

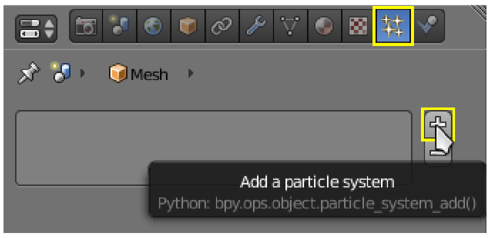


Image 2: Adding a particle system.

To add a new particle system to an object, go to the Particles tab of the object Settings editor and click the small + button. An object can have many Particle Systems.

Each particle system has separate settings attached to it. These settings can be shared among different particle systems, so one doesn't have to copy every setting manually and can use the same effect on multiple objects. Using the Random property they can be randomized to look slightly different, even when using the same settings.

Types of Particle systems

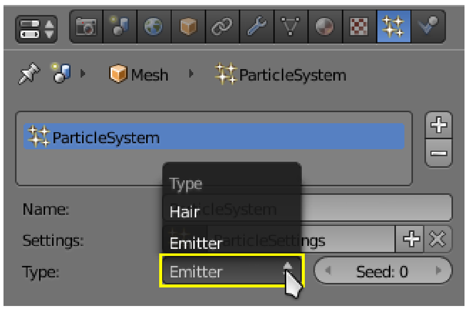


Image 3: Particle system types.

After you have created a particle system, the Property window fills with many panels and buttons. But don't panic! There are two different types of particle systems, and you can change between these two with the Type drop-down list:

Emitter
: This parallels the old system to the greatest extent. In such a system, particles are emitted from the selected object from the Start frame to the End frame and have a certain lifespan.

Hair
: This system type is rendered as strands and has some very special properties: it may be edited in the 3D window in realtime and you can also animate the strands with Cloth Simulation.

The settings in the Particle System panel are partially different for each system type. For example, in *Image 3* they are shown for only system type Emitter.

Common Options

Each system has the same basic sets of controls, but options within those sets vary based on the system employed. These sets of controls are:

Emission	Settings for the initial distribution of particles on the emitter and the way they are born into the scene.
Cache	In order to increase realtime response and avoid unnecessary recalculation of particles, the particle data can be cached in memory or stored on disk.
Velocity	Initial speed of particles.
Rotation	Rotational behavior of particles.
Physics	How the movement of the particles behaves.
Render	Rendering options.
Display	Realtime display in the 3D View.
Children	Control the creation of additional child particles.
Field Weights	Factors for external forces.
Force Field Settings	Makes particles force fields.
Vertex Groups	Influencing various settings with vertex groups.

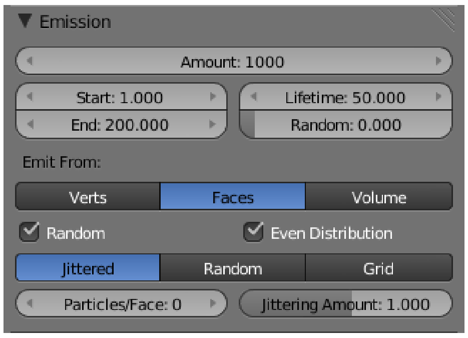
Links

- Tutorials
- Physics Caching and Baking
- Particle Rewrite Documentation
- Thoughts about the particle rewrite code
- Static Particle Fur Library

Particle Emission

The Emitter system works just like its name says: it emits/produces particles for a certain amount of time. In such a system, particles are emitted from the selected object from the Start frame to the End frame and have a certain lifespan. These particles are rendered default as Halos, but you may also render these kind of particles as objects (depending on the particle system's render settings, see Visualization).

Options


Emission

Amount: 1000

Start: 1.000 | Lifetime: 50.000

End: 200.000 | Random: 0.000

Emit From:

Verts | [✓ Faces] | Volume

✓ Random | ✓ Even Distribution

[✓ Jittered] | Random | Grid

Particles/Face: 0 | Jittering Amount: 1.000

Image 2a: Settings for particle Emission.

The buttons in the Emission panel control the way particles are emitted over time:

Amount
: The maximum amount of parent particles used in the simulation.

Start
: The start frame of particle emission. You may set negative values, which enables you to start the simulation before the actual rendering.

End
: The end frame of particle emission.

Lifetime
: The lifetime (in frames) of the particles.

Random
: A random variation of the lifetime of a given particle. The shortest possible lifetime is $Lifetime \times (1-Rand)$. Values above 1.0 are not allowed. For example with the default Lifetime value of 50 a Random setting of 0.5 will give you particles with lives ranging from 50 frames to $50 \times (1.0-0.5)$=25 frames, and with a Random setting of 0.75 you'll get particles with lives ranging from 50 frames to $50 \times (1.0-0.75)$=12.5 frames.

Emission Location

Emit From parameters define how and where the particles are emitted, giving precise control over their distribution. You may use vertex groups to confine the emission, that is done in the Vertexgroups panel.

Verts
: Emit particles from the vertices of a mesh.

Faces
: Emit particles from the surface of a mesh's faces.

Volume
: Emit particles from the volume of an enclosed mesh.

Distribution Settings

These settings control how the emissions of particles are distributed throughout the emission locations

Random
: The emitter element indices are gone through in a random order instead of linearly (one after the other).

For Faces and Volume, additional options appear:

Even Distribution
: Particle distribution is made even based on surface area of the elements, i.e. small elements emit less particles than large elements, so that the particle density is even.

Jittered
: Particles are placed at jittered intervals on the emitter elements.

    Particles/Face

        Number of emissions per face (0 = automatic).

    JitteringAmount

        Amount of jitter applied to the sampling.

Random
: Particles are emitted from random locations in the emitter’s elements.

Grid
: Particles are set in a 3d grid and particles near/in the elements are kept.

  Invert Grid

  : Invert what is considered the object and what is not.

  Hexagonal

  : Uses a hexagonal shaped grid instead of a rectangular one.

  Resolution

  : Resolution of the grid.

  Random

  : Add a random offset to grid locations.

Your mesh must be watertight to emit particles from the volume.

Some modifiers like Edge Split break up the surface, in which case volume emission will not work correctly!

Particle Physics

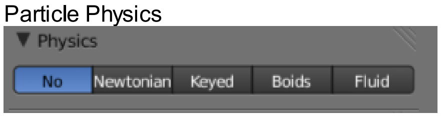


Image 1: Physics types.

The movement of particles may be controlled in a multitude of ways. With particles physics: there are five different systems:

None	It doesn't give the particles any motion, which makes them belong to no physics system.
Newtonian	Movement according to physical laws.
Keyed	Dynamic or static particles where the (animated) targets are other particle systems.
Boids	Particles with limited artificial intelligence, including behavior and rules programming, ideal for flocks of birds or schools of fishes, or predators vs preys simulations.
Fluid	Movement according to fluid laws (based on Smoothed Particle Hydrodynamics technique).

Additional ways of moving particles:

- By softbody animation (only for Hair particle systems).
- By forcefields and along curves.
- By lattices.

Here we will discuss only the particle physics in the narrower sense, i.e. the settings in the Physics panel.

Velocity

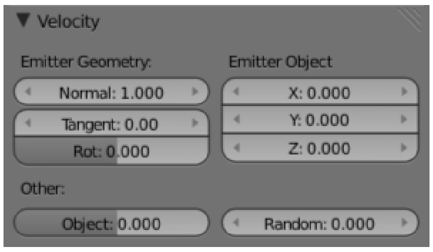


Image 3: Initial velocity.

The initial velocity of particles can be set through different parameters, based on the type of the particle system (see Particle System tab). If the particle system type is Emitter or Hair, then the following parameters give the particle an initial velocity in the direction of…

Emitter Geometry

Normal
: The emitter's surface normals (i.e. let the surface normal give the particle a starting speed).

Tangent

: Let the tangent speed give the particle a starting speed.

Rot
: Rotates the surface tangent.

Emitter Object

Align X,Y,Z
: Give an initial velocity in the X, Y, and Z axes.

Object
: The emitter objects movement (i.e. let the object give the particle a starting speed).

Random
: Gives the starting speed a random variation. You can use a texture to only change the value, see Controlling Emission, Interaction and Time).

Rotation

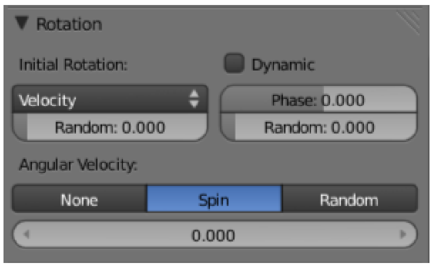


Image 4: Particles rotation.

These parameters specify how the individual particles are rotated during their travel. To visualize the rotation of a particle you should choose visualization type Axis in the Visualization panel and increase the Draw Size.

Initial Rotation Mode
: Sets the initial rotation of the particle by aligning the x-axis in the direction of:

  None
  : the global x-axis.

  Normal
  : Orient to the emitter’s surface normal, the objects Y axis points outwards.

  Normal-Tangent
  : As with normal, orient the Y axis to the surface normal.
    Also orient the X axis to the tangent for control over the objects rotation about the normal. requires UV coordinates, the UV rotation effects the objects orientation, currently uses the active UV layer.
    This allow deformation without the objects rotating in relation to their surface.

  Velocity
  : the particle’s initial velocity.

  Global X/Global Y/Global Z
  : one of the global axes

  Object X/Object Y/Object Z
  : one of the emitter object axes.

  Random
  Randomizes rotation.

Dynamic
: If enabled, only initializes particles to the wanted rotation and angular velocity and let’s physics handle the rest. Particles then change their angular velocity if they collide with other objects (like in the real world due to friction between the colliding surfaces). Otherwise the angular velocity is predetermined at all times (i.e. set rotation to dynamic/constant).

Phase
: Initial rotation phase

Random
: Rand allows a random variation of the Phase.

Angular Velocity
: The magnitude of angular velocity, the dropdown specifies the axis of angular velocity to be

  None
  : a zero vector (no rotation).

  Spin
  : the particles velocity vector.

  Random

  a random vector.

If you use a Curve Guide and want the particles to follow the curve, you have to set Angular Velocity to Spin and leave the rotation on Constant (i.e. don’t turn on Dynamic). Curve Follow does not work for particles.

Common Physics Settings

Size
: Sets the size of the particles.

Random Size
: Give the particles a random size variation.

Mass
: Specify the mass of the particles.

Multiply mass with particle size
: Causes larger particles to have larger masses.

No Physics

At first a Physics type that makes the particles do nothing could seem a bit strange, but it can be very useful at times. None physics make the particles stick to their emitter their whole life time. The initial velocities here are for example used to give a velocity to particles that are effected (or affected?) by a harmonic effector with this physics type when the effect of the effector ends.

Moreover, it can be very convenient to have particles at disposal (whose both Unborn and Died are visible on render) to groom vegetation and/or ecosystems using Object, Group or Billboard types of visualization.

Field Weights

The Field Weight Panel allows you to control how much influence each type of external force field, or effector, has on the particle system. Force fields are external forces that give dynamic systems motion. The force fields types are detailed on the Force Field Page.

Effector Group
: Limit effectors to a specified group. Only effectors in this group will have an effect on the current system.

Gravity
: Control how much the Global Gravity has an effect on the system.

All
: Scale all of the effector weights.

Force Fields

The Force Field Settings Panel allows you to make each individual act as a force field, allowing them to affect other dynamic systems, or even, each other.

Self Effect
: Causes the particle force fields to have an effect on other particles within the same system.

Amount
: Set how many of the particles act as force fields. 0 means all of them are effectors.

You can give particle systems up to 2 force fields. By default they do not have any. Choose an effector type from the dropdowns to enable them. Settings are described on the Force Field Page.

Newtonian Physics

These are the “normal” particle physics. Particles start their life with the specified initial velocities and angular velocities, and move according to Newtonian forces. The response to environment and to forces is computed differently, according to any given integrator chosen by the animator.

Forces

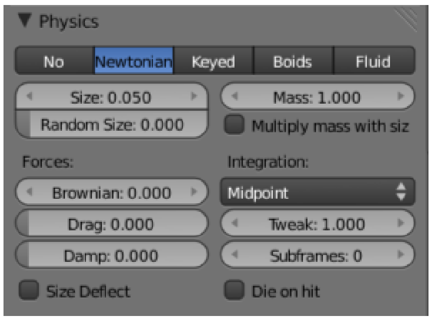


Image 5: Newtonian Physics.

Brownian
: Specify the amount of Brownian motion. Brownian motion adds random motion to the particles based on a Brownian noise field. This is nice to simulate small, random wind forces.

Drag
: A force that reduces particle velocity in relation to it’s speed and size (useful in order to simulate Air-Drag or Water-Drag).

Damp
: Reduces particle velocity (deceleration, friction, dampening).

Collision

Size Deflect
: Use the particle size in deflections.

Die on Hit
: Kill particle when it hits a deflector object.

Integration

Integrators are a set of mathematical methods available to calculate the movement of particles. The following guidelines will help to choose a proper integrator, according to the behavior aimed at by the animator.

Euler	Also known as “Forward Euler”. Simplest integrator. Very fast but also with less exact results. If no dampening is used, particles get more and more energy over time. For example, bouncing particles will bounce higher and higher each time. Should not be confused with “Backward Euler” (not implemented) which has the opposite feature, energies decrease over time, even with no dampening. Use this integrator for short simulations or simulations with a lot of dampening where speedy calculations is more important than accuracy.
Varlet	Very fast and stable integrator, energy is conserved over time with very little numerical dissipation.
Midpoint	Also known as “2nd order Runge-Kutta”. Slower than Euler but much more stable. If the acceleration is constant (no drag for example), it is energy conservative. It should be noted that in example of the bouncing particles, the particles might bounce higher than they started once in a while, but this is not a trend. This integrator is a generally good integrator for use in most cases.
RK4	Short for “4th order Runge-Kutta”. Similar to Midpoint but slower and in most cases more accurate. It is energy conservative even if the acceleration is not constant. Only needed in complex simulations where Midpoint is found not to be accurate enough.

Timestep
: The simulation time step per frame.

Subframes
: Subframes to simulate for improved stability and finer granularity in simulations. Use higher values for faster moving particles.

Keyed Particles

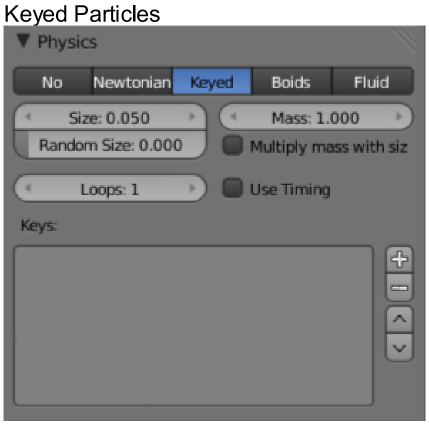


Image 6: Keyed Physics.

The particle paths of keyed particles are determined from the emitter to another particle system's particles. This allows creation of chains of systems with keyed physics to create long strands or groovy moving particles. Basically the particles have no dynamics but are interpolated from one system to the next at drawtime. Because you have so much control over these kind of systems, you may use it

For example, for machines handling fibers (animation of a loom, …). In (Image 3), the strands flow from the bottom system (First keyed) to the second keyed system in the middle, and from that to the top system that has None-Physics. Since you may animate each emitter object as you like, you can do arbitrarily complex animations.

Setup

To setup Keyed particles you need at least two particle systems.

The first system has keyed physics, and it needs the option First activated. This will be the system thats is visible. *The second system may be another keyed system but without the option First, or a normal particle system. This second system is the target of the keyed system.

Loops
: Sets the number of times the keys are looped. Disabled if Use Timing is enabled.

Keys

Key Targets
: You have to enter the name of the object which bears the target system and if there are multiple particle systems the number of the system.

  Click the + to add a key, then select the object.

If you use only one keyed system the particles will travel in their lifetime from the emitter to the target. A shorter lifetime means faster movement. If you have more than one keyed system in a chain, the lifetime will be split equally. This may lead to varying particle speeds between the targets.

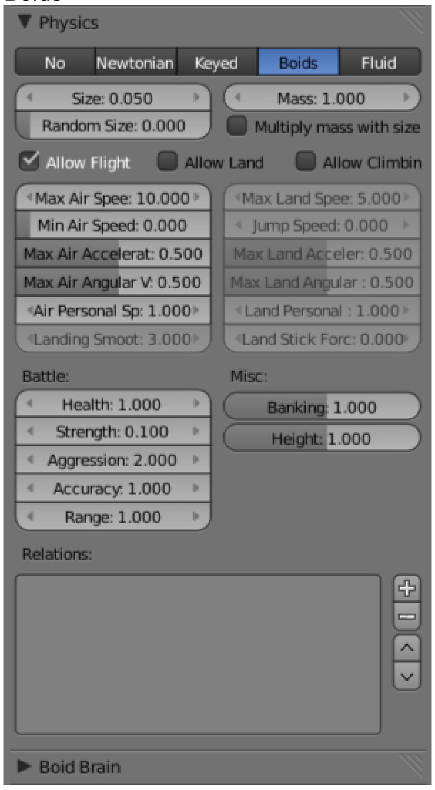
Timing

Use Timing
: Timing works together with the Time slider for the other keyed systems in a chain. The Time slider allows to define a fraction of particle lifetime for particle movement.

An example: let's assume that you have two keyed systems in a chain and a third system as target. The particle lifetime of the first system shall be 50 keys. The particles will travel in 25 frames from the first keyed system to the second, and in further 25 frames from the second system to the target. If you use the Timed button for the first system, the Time slider appears in the second systems panel. It's default value is 0.5, so the time is equally split between the systems. If you set Time to 1, the movement from the first system to the second will get all the lifetime (the particles will die at the second system).

If you set Time to 0 the particles will start at the second system and travel to the target.

Boids



Size: 0.050
Mass: 1.000
Random Size: 0.000
Multiply mass with size
✓ Allow Flight
Allow Land
Allow Climbin
Max Air Spee: 10.000
Max Land Spee: 5.000
Min Air Speed: 0.000
Jump Speed: 0.000
Max Air Accelerat: 0.500
Max Land Acceler: 0.500
Max Air Angular V: 0.500
Max Land Angular : 0.500
Air Personal Sp: 1.000
Land Personal : 1.000
Landing Smoot: 3.000
Land Stick Forc: 0.000
Health: 1.000
Banking: 1.000
Strength: 0.100
Height: 1.000
Aggression: 2.000
Accuracy: 1.000
Range: 1.000

Image 7: Boid Physics.

Boids particle systems can be set to follow basic rules and behaviors. They are useful for simulating flocks, swarms, herds and schools of various kind of animals, insects and fishes. They can react on the presence of other objects and on the members of their own system. Boids can handle only a certain amount of information, therefore the sequence of the Behaviour settings is very important. In certain situations only the first three parameter are evaluated.

To view the subpanel to the right, add a Particle System of type Emitter and look in the middle area of the Particle System tab.

Physics

Boids try to avoid objects with activated Deflection. They try to reach objects with positive Spherical fields, and fly from objects with negative Spherical fields. The objects have to share one common layer to have effect. It is not necessary to render this common layer, so you may use invisible influences.

Boids can different physics depending on whether they are in the air, or on land(on collision object)

Allow Flight
: Allow boids to move in the air.

Allow Land
: Allow boids to move on land.

Allow Climbing
: Allow boids to climb goal objects.

Max Air Speed
: Set the Maximum velocity in the air.

Min Air Speed
: Set the Minimum velocity in the air.

Max Air Acceleration
: Lateral acceleration in air, percent of max velocity (turn). Defines how fast a boid is able to change direction.

Max Air Angular Velocity
: Tangential acceleration in air, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Air Personal Space
: Radius of boids personal space in air. Percentage of particle size.

Landing Smoothness
: How smoothly the boids land.

Max Land Speed
: Set the Maximum velocity on land.

Jump Speed
: Maximum speed for jumping

Max Land Acceleration
: Lateral acceleration on land, percent of max velocity (turn). Defines how fast a boid is able to change direction.

Max Land Angular Velocity
: Tangential acceleration on land, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Land Personal Space
: Radius of boids personal space on land. Percentage of particle size.

Land Stick Force
: How strong a force must be to start effecting a boid on land.

Banking
: Amount of rotation around velocity vector on turns. Banking of (1.0 == natural banking).

Pitch
: Amount of rotation around side vector.

Height
: Boid height relative to particle size.

Battle

Health
: Initial boid health when born.

Strength
: Maximum caused damage per second on attack.

Aggression
: Boid will fight this times stronger than enemy.

Accuracy
: Accuracy of attack.

Range
: Maximum distance of which a boid can attack.

Alliance

The relations box allows you to set up other particle systems to react with the boids. Setting the type to Enemy will cause the systems to fight with each other. Friend will make the systems work together. Neutral will not cause them to align or fight with each other.

Deflectors and Effectors

As mentioned before, very much like Newtonian particles, Boids will react to the surrounding deflectors and fields, according to the needs of the animator:

Deflection: Boids will try to avoid deflector objects according to the Collision rule’s weight. It works best for convex surfaces (some work needed for concave surfaces). For boid physics, Spherical fields define the way the objects having the field are seen by others. So a negative Spherical field (on an object or a particle system) will be a predator to all other boids particle systems, and a positive field will be a goal to all other boids particle systems.

When you select an object with a particle system set on, you have in the Fields tab a little menu stating if the field should apply to the emitter object or to the particle system. You have to select the particle system name if you want prey particles to flew away from predator particles.

Spherical fields: These effectors could be predators (negative Strength) that boids try to avoid or targets (positive Strength) that boids try to reach according to the (respectively) Avoid and Goal rules’ weights. Spherical’s effective Strength is multiplied by the actual relevant weight (e.g. if either Strength or Goal is null, then a flock of boids won’t track a positive Spherical field). You can also activate Die on hit (Extras panel) so that a prey particle simply disappears when “attacked” by a predator particle which reaches it. To make this work, the predator particles have to have a spherical field with negative force, it is not sufficient just to set a positive goal for the prey particles (but you may set the predators force strength to -0.01). The size of the predators and the prey can be set with the Size button in the Extras panel.

Boid Brain

The Boid Brain panel controls how the boids particles will react with each other. The boids' behavior is controlled by a list of rules. Only a certain amount of information in the list can be evaluated. If the memory capacity is exceeded, the remaining rules are ignored.

The rules are by default parsed from top-list to bottom-list (thus giving explicit priorities), and the \order can be modified using the little arrows buttons on the right side.

The list of rules available are:

Goal
: Seek goal (objects with Spherical fields and positive Strength)

    Predict
    : Predict target's movements

Avoid
: Avoid “predators” (objects with Spherical fields and negative Strength)

    Predict
    : Predict target's movements

    Fear Factor
    : Avoid object if danger from it is above this threshold

Avoid Collision
: Avoid objects with activated Deflection

    Boids

Avoid collision with other boids

Deflectors
: Avoid collision with deflector objects

Look Ahead
: Time to look ahead in seconds

Separate
: Boids move away from each other

Flock
: Copy movements of neighboring boids, but avoid each other

Follow Leader
: Follows a leader object instead of a boid

    Distance
    : Distance behind leader to follow

    Line
    : Follow the leader in a line

Average Speed
: Maintain average velocity.

    Speed
    : Percentage of maximum speed

    Wander
    : How fast velocity's direction is randomized

    Level
    : How much velocity's Z component is kept constant

Fight
: Move toward nearby boids

    Fight Distance
    : Attack boids at a maximum of this distance

    Flee Distance
    : Flee to this distance

Rule Evaluation

There are three ways control how rules are evaluated.

Average
: All rules are averaged.

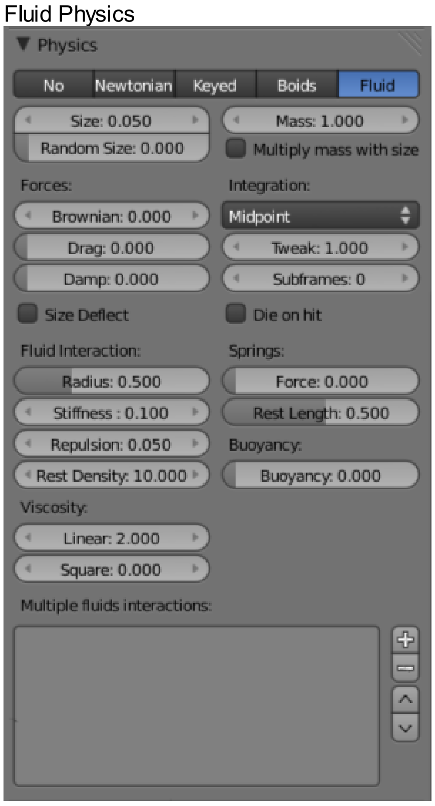
Random
: A random rule is selected for each boid.

Fuzzy
: Uses fuzzy logic to evaluate rules. Rules are gone through top to bottom. Only the first rule that effect above fuzziness threshold is evaluated. The value should be considered how hard the boid will try to respect a given rule (a value of 1.000 means the Boid will always stick to it, a value of 0.000 means it will never). If the boid meets more than one conflicting condition at the same time, it will try to fulfill all the rules according to the respective weight of each.

Please note that a given boid will try as much as it can to comply to each of the rules he is given, but it is more than likely that some rule will take precedence on other in some cases. For example, in order to avoid a predator, a boid could probably “forget” about Collision, Crowd and Center rules, meaning that “while panicked” it could well run into obstacles, for example, even if instructed not to, most of the time.

As a final note, the Collision algorithm is still not perfect and in research progress, so you can expect wrong behaviors at some occasion. It is worked on.

Fluid Physics



Size: 0.050
Mass: 1.000
Random Size: 0.000
Multiply mass with size

Forces:
Brownian: 0.000
Drag: 0.000
Damp: 0.000
Size Deflect

Integration:
Midpoint
Tweak: 1.000
Subframes: 0
Die on hit

Fluid Interaction:
Radius: 0.500
Stiffness : 0.100
Repulsion: 0.050
Rest Density: 10.000

Springs:
Force: 0.000
Rest Length: 0.500

Buoyancy:
Buoyancy: 0.000

Viscosity:
Linear: 2.000
Square: 0.000

Multiple fluids interactions:

Image 8: Fluid Physics.

Fluid simulations are widely used in CG, and a very desired feature of any particle system, fluid particles are similar to newtonian ones but this time particles are influenced by internal forces like pressure, surface tension, viscosity, springs, etc. Blender particle fluids use the SPH techniques to solve the particles fluid equations.

Smoothed-particle hydrodynamics (SPH) is a computational method used for simulating fluid flows. It has been used in many fields of research, including astrophysics, ballistics, vulcanology, and oceanography. It is a mesh-free Lagrangian method (where the co-ordinates move with the fluid), and the resolution of the method can easily be adjusted with respect to variables such as the density.

From liquids to slime, goo to sand and wispy smoke the possibilities are endless.

Settings

Fluid physics share options with Newtonian Physics. These are covered on that page.

Fluid Properties

Stiffness
: How incompressible the fluid is.

Viscosity
: Linear viscosity. Use lower viscosity for thicker fluids.

Buoyancy
: Artificial buoyancy force in negative gravity direction based on pressure differences inside the fluid.

Advanced

Repulsion Factor
: How strongly the fluid tries to keep from clustering (factor of stiffness). Check box sets repulsion as a factor of stiffness.

Stiff Viscosity
: Creates viscosity for expanding fluid. Check box sets this to be a factor of normal viscosity.

Interaction Radius
: Fluid's interaction radius. Check box sets this to be a factor of 4*particle size.

Rest Density
: Density of fluid when at rest. Check box sets this to be a factor of default density.

Springs

Force
: Spring force

Rest Length
: Rest length of springs. Factor of particle radius. Check box sets this to be a factor of 2*particle size.

Viscoelastic Springs
: Use viscoelastic springs instead of Hooke's springs.

Elastic Limit
: How much the spring has to be stretched/compressed in order to change its rest length

Plasticity
: How much the spring rest length can change after the elastic limit is crossed.

Initial Rest Length
: Use initial length as spring rest length instead of 2*particle size.

Frames
: Create springs for this number of frames since particle's birth (0 is always).

Particle Visualization

With the items in the Display and Render panel you can set the way the particles will be rendered or depicted in the view ports in various ways. Some option are valid only for the 3D window, the particles then are rendered always as Halos. Some of the options will be rendered as shown in the 3D window.

Viewport Display

The Display Panel controls how particles are displayed in the 3d viewport. This does not necessarily determine how they will appear when rendered.

None
: The particles are not shown in the 3D window and are not rendered. The emitter may be rendered though.

Point
: Particles are displayed as square points. Their size is independent of the distance from the camera.

Circle
: Particles are displayed as circles that face the view. Their size is independent of the distance from the camera.

Cross
: Particles are displayed as 6-point crosses that align to the rotation of the particles. Their size is independent of the distance from the camera.

Axis
: Particles are displayed as 3-point axes. This useful if you want to see the orientation and rotation of particles in the view port. Increase the Draw Size until you can clearly distinguish the axis.

Particles visualized like Point, Circle, Cross and Axis don’t have any special options, but can be very useful when you have multiple particle systems at play, if you don’t want to confuse particles of one system from another (e.g. in simulations using Boids physics).

Display
: Specifies the percentage of all particles to show in the viewport (all particles are still rendered).

Draw Size
: Specifies how large (in pixels) the particles are drawn in the viewport (0 = default).

Size
: Draw the size of the particles with a circle.

Velocity
: Draw the velocity of the particles with a line that points in the direction of motion, and length relative to speed.

Number
: Draw the id-numbers of the particles in the order of emission.

Color

The Color Menu allows you to draw particles according to certain particle properties.

None
: Particles are black.

Material
: Particles are colored according to the material they are given.

Velocity
: Color particles according to their speed. The color is a ramp from blue to green to red, Blue being the slowest, and Red being velocities approaching the value of Max or above. Increasing Max allows for a wider range of particle velocities.

Acceleration
: Color particles according to their acceleration.

Render Settings

The Render Panel controls how particles appear when they are rendered.

Material Index
: Set which of the object's material is used to shade the particles.

Parent
: Use a different object's coordinates to determine the birth of particles.

Emitter
: When disabled, the emitter is no longer rendered. Activate the button Emitter to also render the mesh.

Parents
: Render also parent particles if child particles are used. Children have a lot of different deformation options, so the straight parents would stand between their curly children. So by default Parents are not rendered if you activate Children.. See Children

Unborn
: Render particles before they are born.

Died
: Render particles after they have died. This is very useful if particles die in a collision (Die on hit), so you can cover objects with particles.

None

When set to None particles are not rendered. This is useful if you are using the particles to duplicate objects.

Halo

Halo particles are rendered as Halo Type Materials.

Trail Count
: Set the number of trail particles. When greater than 1, additional options appear.

Length in Frames
: Path timing is in absolute frames.

Length
: End time of drawn path.

Random
: Give path lengths a random variation.

Line

The Line visualization mode creates (more or less thin) polygon lines with the strand renderer in the direction of particles velocities. The thickness of the line is set with the parameter Start of the Strands shader (Material sub-context, Links and Pipeline panel).

Back
: Set the length of the particle's tail.

Front
: Set the length of the particle's head.

Speed
: Multiply the line length by particles' speed. The faster, the longer the line.

Trail Count
: See description in the Halo Render Type above.

Path

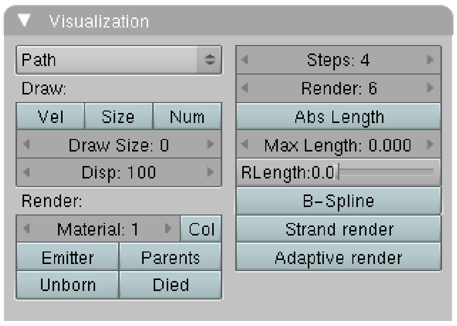


Image 3: The Visualization panel for Path visualization.

The Path visualization needs a Hair particle system or Keyed particles.

Strand render
: [Keypointstrands] Use the strand primitive for rendering. Very fast and effective renderer.

Adaptive render
: Tries to remove unnecessary geometry from the paths before rendering particle strands in order to make the render faster and easier on memory.

Angle
: How many degrees path has to curve to produce another render segment (straight parts of paths need fewer segments).

Pixel
: How many pixels path has to cover to produce another render segment (very short hair or long hair viewed from far away need fewer parts). (only for Adaptive render).

B-Spline
: Interpolate hair using B-Splines. This may be an option for you if you want to use low Render values. You loose a bit of control but gain smoother paths.

Steps
: Set the number of subdivisions of the rendered paths (the value is a power of 2). You should set this value carefully, because if you increase the render value by two you need four times more memory to render. Also the rendering is faster if you use low render values (sometimes drastically). But how low you can go with this value depends on the waviness of the hair.(the value is a power of 2). This means 0 steps give 1 subdivision, 1 give 2 subdivisions, 2→4, 3→8, 4→16, … $n \rightarrow 2^n$.

Timing Options:

Absolute Path Time
: Path timing is in absolute frames.

Start
: Start time of the drawn path.

End
: End time of the drawn path.

Random
: Give the path length a random variation.

Please see also the manual page about Strands for an in depth description.

Object

In the Object visualization mode the specified object (Dupli Object: field) is duplicated in place of each particle. The duplicated object has to be at the center of the coordinate system, or it will get an offset to the particle.

Global
: Use object's global coordinates for duplication.

Size
: Size of the objects

Random Size
: Give the objects a random size variation.

Group

In the Group visualization mode, the objects that belong to the group (GR: field) are duplicated sequentially in the place of the particles.

WholeGroup
: Use the whole group at once, instead of one of its elements, the group being displayed in place of each particle.

Use Count
: Use objects multiple times in the same groups. Specify the order and nuber of times to repeat each object with the list box that appears. You can duplicate an object in the list with the + button, or remove a duplicate with the - button.

Use Global
: Use object's global coordinates for duplication.

Pick Random
: The objects in the group are selected in a random order, and only one object is displayed in place of a particle.
: Please note that this mechanism fully replaces old Blender particles system using parentage and DupliVerts to replace particles with actual geometry. This method is fully deprecated and doesn't work anymore.

Size
: Size of the objects

Random Size
: Give the objects a random size variation.

Billboard

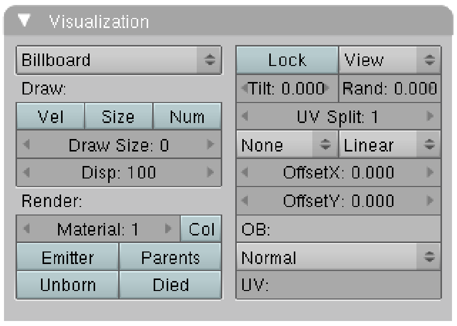


Image 4: Billboard visualization for particles.

Billboards are aligned square planes. They are aligned to the camera by default, but you can choose another object that they should be aligned to.

If you move a billboard around it's target, it always faces the center of it's target. The size of a billboard is set with the parameter Size of the particle (in Blender Units). You can use them e.g. for Sprites, or to replace Halo visualization. Everything that can be done with a halo can also be done with a billboard. But billboards are real objects, they are seen by raytracing, they appear behind transparent objects, they may have an arbitrary form and receive light and shadows. They are a bit more difficult to set up and take more render time and resources.

Texturing billboards (including animated textures with alpha) is done by using uv coordinates that are generated automatically for them so they can take an arbitrary shape. This works well for animations, because the alignment of the billboards can be dynamic. The textures can be animated in several ways:

- Depending on the particle lifetime (relative time).
- Depending on the particle starting time.
- Depending on the frame (absolute time).

You can use different sections of an image texture:

- Depending on the lifetime of the billboard.
- Depending on the emission time.
- Depending on align or tilt.

Since you use normal materials for the billboard you have all freedoms in mixing textures to your liking. The material itself is animated in absolute time.

The main thing to understand is that if the object doesn't have any *UV Layers*, you need to create at least one in the objects Editing context, for any of these to work. Moreover, the texture has to be set to UV coordinates in the Map Input panel. If you want to see examples for some of the animation possibilities, see the Billboard Animation Tutorial.

An interesting alternative to billboards are in certain cases strands, because you can animate the shape of the strands. Because this visualization type has so much options it is explained in greater detail below.

You can limit the movement with these options. How the axis is prealigned at emission time.

View
: No prealignement, normal orientation to the target.

X/Y/Z
: Along the global X/Y/Z-axis respectively.

Velocity
: Along the speed vector of the particle.

Lock
: Locks the align axis, keeps this orientation, the billboard aligns only along one axis to it's target

Billboard Object
: The target object that the billboards are facing. By default, the active camera is used.

Tilt Angle
: Rotation angle of the billboards planes. A tilt of 1 rotates by 180 degrees (turns the billboard upside down).

Random
: Random variation of tilt.

Offset X
: Offset the billboard horizontally in relation to the particle center, this does not move the texture.

Offset Y
: Offset the billboard vertically in relation to the particle center.

UV Channels
: Billboards are just square polygons. To texture them in different ways we have to have a way to set what textures we want for the billboards and how we want them to be mapped to the squares. These can then be set in the texture mapping buttons to set wanted textures for different coordinates. You may use three different UV layers and get three different sets of UV coordinates, which can then be applied to different (or the same) textures.

Billboard Normal UV
: Coordinates are the same for every billboard, and just place the image straight on the square.

Billboard Time-Index (X-Y)
: Coordinates actually define single points in the texture plane with the x-axis as time and y-axis as the particle index. For example using a horizontal blend texture mapped to color from white to black will give us particles that start off as white and gradually change to black during their lifetime. On the other hand a vertical blend texture mapped to color from white to black will make the first particle to be white and the last particle to be black with the particles in between a shade of gray.

The animation of the UV textures is a bit tricky. The UV texture is split into rows and columns (N times N). The texture should be square. You have to use UV Split in the UV channel and fill in the name of the UV layer. This generated UV coordinates for this layer.

Split UV's
: The amount of rows/columns in the texture to be used.
: Coordinates are a single part of the UV Split grid, which is a n×n grid over the whole texture. What the part is used for each particle and at what time is determined by the Offset and Animate controls. These can be used to make each billboard unique or to use an “animated” texture for them by having each frame of the animation in a grid in a big image.

Billboard Split UV
: Set the name of the *UV layer* to use with billboards (you can use a different one for each UV Channel). By default, it is the active UV layer (check the Mesh panel in the Editing context, F9).

Animate
: Dropdown menu, indicating how the split UVs could be animated (changing from particle to particle with time):

    None

        No animation occurs on the particle itself, the billboard uses one section of the texture in it's lifetime.

    Age

        The sections of the texture are gone through sequentially in particles' lifetimes.

    Angle

        Change the section based on the angle of rotation around the Align to axis, if View is used the change is based on the amount of tilt.

    Frame

        The section is changes according to the frame.

Offset
: Specifies how to choose the first part (of all the parts in the n×n grid in the texture defined by the UV Split number) for all particles.

None

All particles start from the first part.

Linear

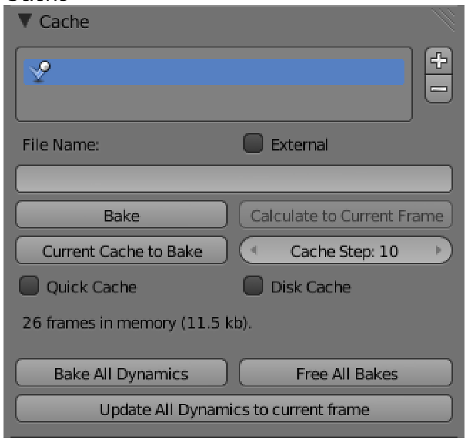
First particle will start from the first part and the last particle will start from the last part, the particles in between will get a part assigned linearly from the first to the last part.

Random

Give a random starting part for every particle.

Trail Count
See the description in the Halo Render Type above.

Cache



26 frames in memory (11.5 kb).

Image 4: Cache panel for particles.

Emitter systems use a unified system for caching and baking (together with softbody and cloth). The results of the simulation are automatically cached to disk when the animation is played, so that the next time it runs, it can play again quickly by reading in the results from the disk. If you Bake the simulation the cache is protected and you will be asked when you're trying to change a setting that will make a recalculating necessary.

Beware of the Start and End Settings

The simulation is only calculated for the positive frames in-between the Start and End frames of the Bake panel, whether you bake or not. So if you want a simulation longer than 250 frames you have to change the End frame!

Caching

- As animation is played, each physics system writes each frame to disk, between the simulation start and end frames. These files are stored in folders with prefix "`blendcache`", next to the .blend file. Note that for the cache to fill up, one has to start playback before or on the frame that the simulation starts.
- The cache is cleared automatically on changes - but not on all changes, so it may be necessary to free it manually e.g. if you change a force field.
- If it is impossible to write in the subdirectory there will be no caching.
- The cache can be freed per physics system with a button in the panels, or with the CtrlB shortcut key to free it for all selected objects.
- If the file path to the cache is longer than what is possible with your operating system (more than 250 characters for example), strange things might happen.

Baking

- The system is protected against changes after baking.
- The Bake result is cleared also with CtrlB for all selected objects or click on Free Bake for a singular particle system.
- If the mesh changes the simulation is not calculated anew.
- Sorry: no bake editing for particles like for softbodies and clothes.

Two notes at the end:

- For renderfarms, it is best to bake all the physics systems, and then copy the blendcache to the renderfarm as well.
- Be careful with the sequence of modifiers in the modifier stack (as always). You may have a different number of faces in the 3D window and for rendering (e.g. when using subdivision surface), if so, the rendered result may be very different from what you see in the 3D window.

Hair

When set to hair mode, particle system creates only static particles, which may be used for hair, fur, grass and the like.

Growing

The first step is to create the hair, specifying the amount of hair strands and their lengths.

The complete path of the particles is calculated in advance. So everything a particle does a hair may do also. A hair is as long as the particle path would be for a particle with a lifetime of 100 frames. Instead of rendering every frame of the particle animation point by point there are calculated control points with an interpolation, the segments.

Styling

The next step is to style the hair. You can change the look of base hairs by changing the Physics Settings.

A more advanced way of changing the hair appearance is to use Children. This adds child hairs to the original ones, and has settings for giving them different types of shapes.

You can also interactively style hairs in Particle Mode. In this mode, the particle settings become disabled, and you can comb, trim, lengthen, etc. the hair curves.

Animating

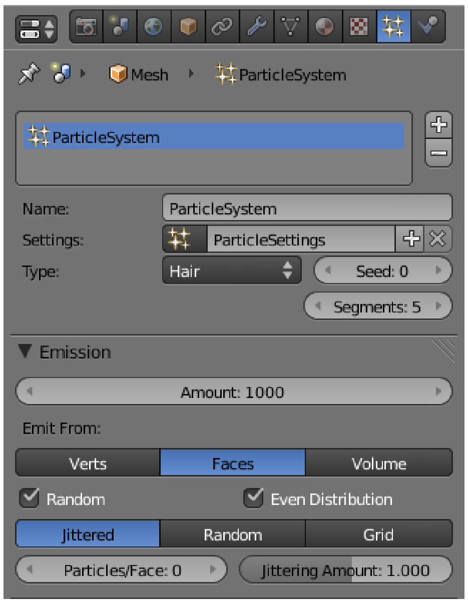
Hair can now be made dynamic using the cloth solver. This is covered in the Hair Dynamics page.

Rendering

Blender can render hairs in several different ways. Materials have a Strand section, which is covered in the materials section in the Strands Page.

Hair can also be used as a basis for the Particle Instance modifier, which allows you to have a mesh be deformed along the curves, which is useful for thicker strands, or things like grass, or feathers, which may have a more specific look.

Options



Name: ParticleSystem
Settings: ParticleSettings
Type: Hair
Seed: 0
Segments: 5
Amount: 1000
Emit From: Faces
✓ Random
✓ Even Distribution
Particles/Face: 0
Jittering Amount: 1.000

Image 4a: Settings for a Hair particle system.

Regrow
: Regrow Hair for each frame.

Advanced
: Enables advanced settings which reflect the same ones as working in Emitter mode.

Emission

Amount
: Set the amount of hair strands. Use as little particles as possible, especially if you plan to use softbody animation later. But you need enough particles to have good control. For a “normal” haircut I found some thousand (very roughly 2000) particles to give enough control. You may need a lot more particles if you plan to cover a body with fur. Volume will be produced later with Children.

Hair Dynamics

Settings for adding movement to hair see Hair Dynamics.

Display

Rendered
: Draw hair as curves.

Path
: Draw just the end points if the hairs.

Steps
: The number of segments (control points minus 1) of the hair strand. In between the control points the segments are interpolated. The number of control points is important:

- for the softbody animation, because the control points are animated like vertices, so more control points mean longer calculation times.
- for the interactive editing, because you can only move the control points (but you may recalculate the number of control points in Particle Mode).

10 Segments should be sufficient even for very long hair, 5 Segments are enough for shorter hair, and 2 or 3 segments should be enough for short fur.

Children

See Children.

Render

Hair can be rendered as a Path, Object, or Group. See Particle Visualization for descriptions.

Usage

Image 4b: Particle systems may get hairy…

- Fur Tutorial, which produced (*Image 4b*). It deals especially with short hair.

- Blender Hair Basics, a thorough overview of all of the hair particle settings.

Hair Dynamics

Hair particles can now be made dynamic using Cloth physics.

To enable hair physics, click the check box beside Hair Dynamics.

Settings

Material

Stiffness
: Controls how stiff the root of the hair strands are.

Mass
: Controls the mass of the cloth material.

Bending
: Controls the amount of bend along the hairs. Higher values cause less bending.

Internal Friction
: Amount of friction between individual hairs.

Collider Friction
: Amount of friction between hairs and external collision objects.

Damping

Spring
: Damping of cloth velocity. (higher = more smooth, less jiggling).

Air
: Air has normally some thickness which slows falling things down.

Quality

Steps
: Quality of the simulation in steps per frame. (higher is better quality but slower).

Children

Children are Hair and Keyed particles assigned subparticles. They make it possible to work primarily with a relatively low amount of Parent particles, for whom the physics are calculated. The children are then aligned to their parents. Without recalculating the physics the number and visualization of the children can be changed.

- Children can be emitted from particles or from faces (with some different options). Emission from Faces has some advantages, especially the distribution is more even on each face (which makes it better suitable for fur and the like). However, children from particles follow their parents better, e.g. if you have a softbody animation and don't want the hair to penetrate the emitting mesh. But see also our manual page about Hair.
- If you turn on children the parents are no longer rendered (which makes sense because the shape of the children may be quite different from that of their parents). If you want to see the parents additionally turn on the Parents button in the Visualization panel.
- Children carry the same material as their parents and are colored according to the exact place from where they are emitted (so all children may have different color or other attributes).

The possible options depend from the type of particle system, and if you work with *Children from faces* or *Children from particles*. We don't show every possible combination, only the settings for a Hair particle system.

Settings

Simple
: Children are emitted from the parent hairs.

Interpolated
: Children are emitted between the *Parent* particles on the faces of a mesh. They interpolate between adjacent parents. This is especially useful for fur, because you can achieve an even distribution. Some of the children can become virtual parents, which are influencing other particles nearby.

Display
: The number of children in the 3D window.

Render
: The number of children to be rendered (up to 10.000).

For Simple Mode

Size
: Only for Emitter. A multiplier for children size.

Random
: Random variation to the size of child particles.

Interpolated Mode

Seed
: Offset the random number table for child particles, to get a different result.

Virtual
: Relative amount of virtual parents.

Long Hair
: Calculate children that suit long hair well.

Effects

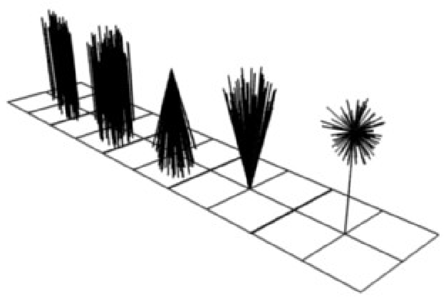


Image 2: From left to right: Round: 0.0 / Round: 1.0 /
Clump: 1.0 / Clump: -1.0 / Shape: -0.99.

Clump
: Clumping. The children may meet at their tip (1.0) or start together at their root (-1.0).

Shape
: Form of Clump. Either inverse parabolic (0.99) or exponentially (-0.99).

Length
: Length of child paths

Threshold
: Amount of particles left untouched by child path length

Radius
: The radius in which the children are distributed around their parents. This is 3D, so children may be emitted higher or lower than their parents.

Roundness
: The roundness of the children around their parents. Either in a sphere (1.0) or in-plane (0.0).

Seed
: Offset in the random number table for child particles, to get a different randomized result

Roughness

Uniform,Size
: It is based on children location so it varies the paths in a similar way when the children are near.

Endpoint,Shape
: “Rough End” randomizes path ends (a bit like random negative clumping). Shape may be varied from <1 (parabolic) to 10.0 (hyperbolic).

Random,Size,Threshold
: It is based on a random vector so it’s not the same for nearby children. The threshold can be specified to apply this to only a part of children. This is useful for creating a few stray children that won’t do what others do.

Kink

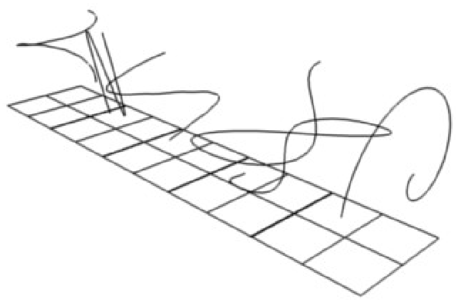


Image 3: Child particles with Kink. From left to right: Curl / Radial / Wave / Braid / Roll.

With Kink you can rotate the children around the parent. See above picture (*Image 3*) for the different types of Kink.

Curl
: Children grow in a spiral around the parent hairs.

Radial
: Children form around the parent a wave shape that passes through the parent hair.

Wave
: Children form a wave, all in the same direction.

Braid
: Children braid themselves around the parent hair.

Amplitude
: The amplitude of the offset.

Clump
: How much clump effects kink amplitude.

Flatness
: How flat the hairs are.

Frequency
: The frequency of the offset (1/total length). The higher the frequency the more rotations are done.

Shape
: Where the rotation starts (offset of rotation).

Vertex Groups

The Vertexgroups panel allows you to specify vertex groups to use for several child particle settings. You can also negate the effect of each vertex group with the check boxes. You can affect the following attributes:

- Density
- Length
- Clump
- Kink
- Roughness 1
- Roughness 2
- Roughness End

Examples

...

Particle Mode

Using Particle Mode you can edit the key-points (key-frames) and paths of Baked Hair, Particle, Cloth, and Soft Body simulations. (You can also edit and style hair before baking).

Since working in particle mode is pretty easy and very similar to working with vertices in the 3D window, we will show how to set up a particle system and then give a reference of the various functions.

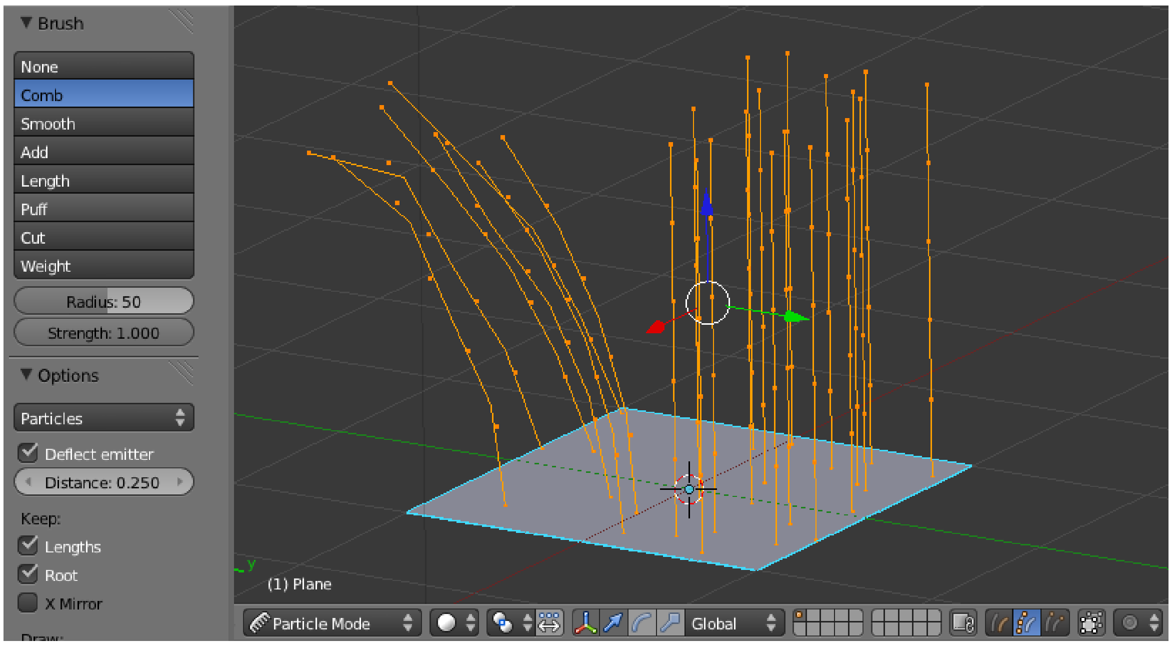
Ways to use Particle Mode

Only Frames Baked to Memory are Editable!

If you cannot edit the particles, check that you are not baking to a Disk Cache.

Setup for Hair Particles

- Create a Hair particle system - With your object selected, click the Particle System icon in the Properties panel. Create a new particle system by clicking the Plus.
- Give it an initial velocity in the Normal direction (first check the Advanced box, then modify the Velocity sub-panel), or adjust the Hair Length.
- Create a simulation - Place the camera at a good position (» View » Cameras » Active Camera ... or 0 NumPad). Check the Hair Dynamics box. Select » Render » Render OpenGL Animation in Render Engine mode.



Editing hair strands in Particle Mode

Editing a baked particle simulation's particle paths in Particle Mode

Setup for Particle, Cloth, and Soft Body Simulations

- Use Emitter particles, or a cloth/soft-body simulation
- Create a simulation - set up objects and or emitters, set your time range (use a small range if you are just starting out and experimenting), set up the simulation how you want it, using Alt+a to preview it.

Bake the Simulation

- Once you are happy with the general simulation, bake the simulation from object mode. The simulation must be baked to enable editing. (remember to bake to memory, a disk cache will not be editable in Particle Mode)

Edit the Simulation

- Switch to Particle Edit from the Mode dropdown menu in the bottom menu bar of the 3D View to edit the particle's paths/key-frames. You may need to press T from within the 3D viewport to see the Particle Edit panel. Move to the frame you want to edit and use the various Particle Edit tools to edit your simulation. Work slowly, previewing your changes with Alt+a, and save often so that you can go back to the previous version should something happen, or that you do not like the latest changes you have made.

To be able to clearly see what you are working on:

- Turn on the Particle Edit Properties (*PEP*) panel with N.
- Select Point select mode
  in the header of the 3D window. This will display key points along the particle path.

Brush Size

Press F to resize the brush while working

Using Particle Mode

Selecting Points

- Single: RMB .
- All: A.
- Linked: Move the mouse over a keypoint and press L.
- Border select: B.
- First/last: W → Select First/Select Last.

You may also use the Select Menu.

Selections

Selections are extremely useful for modifying only the particles that you want. Hover over a particle path and press L to link-select it, hover over the next and press L to add that path to the selection. To remove a path, hold shift and press L. To Deselect all press A. The method to select individual points is the same as in edit mode. click to select, shift+click to add/remove a point from the selection

Beware of Undo!

Using Undo in Particle Mode can have strange results. Remember to save often!

Moving keypoints or particles

- To move selected keypoints press G, or use one of the various other methods to grab vertices.
- To move a particle root you have to turn off Keep Root in the Tool Bar.
- You can do many of the things like with vertices, including scaling, rotating and removing (complete particles or single keys).
- You may not duplicate or extrude keys or particles, but you can subdivide particles which adds new keypoints (W → Subdivide/2 NumPad).
- Alternatively you can rekey a particle (W → Rekey/1 NumPad) and choose the number of keys.

How smoothly the hair and particle paths are displayed depends on the Path Steps setting in the Tool Bar. Low settings produce blocky interpolation between points, while high settings produce a smooth curve.

Mirroring particles

- If you want to create an X-Axis symmetrical haircut you have to do following steps:
  - Select all particles with A.
  - Mirror the particles with CtrlM, or use the Particle → Mirror menu.
  - Turn on X-Axis Mirror Editing in the Particle menu.

It may happen that after mirroring two particles occupy nearly the same place. Since this would be a waste of memory and rendertime, you can Remove doubles either from the Specials (W) or the Particle menu.

Hiding/Unhiding

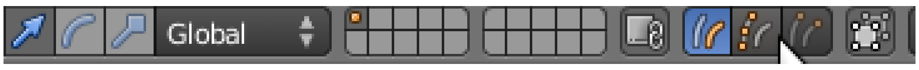
Hiding and unhiding of particles works similar as with vertices in the 3D window. Select one or more keypoints of the particle you want to hide and press H. The particle in fact doesn't vanish, only the key points.

Hidden particles (i.e. particles whose keypoints are hidden) don't react on the various brushes. But:

If you use Mirror Editing even particles with hidden keypoints may be moved, if their mirrored counterpart is moved.

To un-hide all hidden particles press Alt+H.

Select Modes



Path
: No keypoints are visible, you can select/deselect only all particles.

Point
: You see all of the keypoints.

Tip
: You can see and edit (including the brushes) only the tip of the particles, i.e. the last keypoint.

Brush

With the buttons you can select the type of “Comb” utility you want to use. Below the brush types, their settings appear

Common Options:

Radius
: Set the radius if the brush.

Strength
: Set the strength of the brush effect (not for Add brush).

Add/Sub Grow/Shrink
: Sets the brush to add the effect or reverse it..

None
: No special tool, just edit the keypoints as “normal” vertices.

Comb
: Moves the keypoints (similar to “proportional editing”).

Smooth
: Parallels visually adjacent segments.

Add
: Adds new particles.

Count
: The number of new particles per step.

Interpolate
: Interpolate the shape of new hairs from existing ones.

Steps
: Amount of brush steps

Keys
: How many keys to make new particles with.

Length
: Scales the segments, so it makes the hair longer(Grow) or shorter(Shrink).

Puff
: Rotates the hair around it’s first keypoint (root). So it makes the hair stand up (Add) or lay down (Sub).

Puff Volume
: Apply puff to unselected end-points, (helps maintain hair volume when puffing root)

Cut
: Scales the segments until the last keypoint reaches the brush.

Weight
: This is especially useful for softbody animations, because the weight defines the softbody Goal. A keypoint with a weight of 1 won’t move at all, a keypoint with a weight of 0 subjects fully to softbody animation. This value is scaled by the GMin-GMax range of softbody goals.

Options

Deflect Emitter,Dist
: Don’t move keypoints through the emitting mesh. Dist is the distance to keep from the Emitter.

Keep
: Length

Keep the length of the segments between the keypoints when combing or smoothing the hair. This is done by moving all the other keypoints.

Root

Keep first key unmodified, so you can’t transplant hair.

X Mirror
: Enable mirror editing across the local x axis.

Draw
: Path Steps

  Drawing steps, sets the smoothness of the drawn path.

  Show Children
  Draws the children of the particles too. This allows to fine tune the particles and see their effects on the result, but it may slow down your system if you have many children.

Soft Bodies

Image 1a: A softbody cloth uncovering a text. Animation – Blend file

A Soft Body in general, is a simulation of a soft or rigid deformable object. In Blender, this system is best for simple cloth objects and closed meshes. There is dedicated Cloth Simulation physics that use a different solver, and is better for cloth.

This simulation is done by applying forces to the vertices or controlpoints of the object. There are exterior forces like gravity or forcefields and interior forces that hold the vertices together. This way you can simulate the shapes that an object would take on in reality if it had volume, was filled with something, and was acted on by real forces.

Soft Bodies can interact with other objects (*Collision*). They can interact with themselves (*Self Collision*).

The result of the Soft Body simulation can be converted to a static object. You can also *bake edit* the simulation, i.e. edit intermediate results and run the simulation from there.

Typical scenarios for using Soft Bodies

Image 1b: A wind cone. The cone is a Soft Body, as the suspension. Animation – Blend file

Soft Bodies are well suited for:

- Elastic objects with or without collision.
- Flags, fabric reacting to forces.
- Certain modeling tasks, like a cushion or a table cloth over an object.
- Blender has another simulation system for clothing (see Clothes). But you can sometimes use Soft Bodies for certain parts of clothing, like wide sleeves.
- Hair (as long as you minimize collision).
- Animation of swinging ropes, chains and the like.

The following videos may give you some more ideas: [1], [2]

Creating a Soft Body

Soft Body simulation works for all objects that have vertices or control points:

- Meshes.
- Curves.
- Surfaces.
- Lattices.

To activate the Soft Body simulation for an object:

- In the Properties window, go to the Physics tab (it is all the way on the right, and looks like a bouncing ball).
- Activate the Soft Body button.

A lot of options appear. For a reference of all the settings see this page.

- You start a Soft Body simulation with AltA.
- You pause the simulation with Space, continue with AltA.

- You stop the simulation with Esc.

Simulation Quality

The settings in the Soft Body Solver panel determine the accuracy of the simulation.

Min Step
: Minimum simulation steps per frame. Increase this value, if the Soft Body misses fast moving collision objects.

Max Step
: Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the Error Limit) but you have probably a good reason to change it.

Auto-Step
: Use Velocities for automatic step sizes.

Error Limit
: Rules the overall quality of the solution delivered. Default 0.1. The most critical setting that says how precise the solver should check for collisions. Start with a value that is 1/2 the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how “bad” it is doing and the Error Limit causes the solver to do some “adaptive step sizing”.

Fuzzy
: Simulation is faster, but less accurate.

Choke
: Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Diagnostics

Print Performance to Console
: Prints on the console how the solver is doing.

Estimate Matrix
: Estimate matrix. Split to COM , ROT ,SCALE

Cache and Bake

Soft Bodies and other physic simulations use a unified system for caching and baking. See Particle Cache for reference.

The results of the simulation are automatically cached to disk when the animation is played, so that the next time it runs, it can play again quickly by reading in the results from the disk. If you Bake the simulation the cache is protected and you will be asked when you’re trying to change a setting that will make a recalculating necessary.

Beware of the Start and End settings

The simulation is only calculated for the frames in-between the Start and End frames (Bake panel), even if you don’t actually bake the simulation! So if you want a simulation longer than the default setting of 250 frames you have the change the End frame.

- Caching:
  - As animation is played, each physics system writes each frame to disk, between the simulation start and end frames. These files are stored in folders with prefix “`blendcache`”, next to the .blend file.
  - The cache is cleared automatically on changes - but not on all changes, so it may be necessary to free it manually, e.g. if you change a force field. Note that for the cache to fill up, one has to start playback before or on the frame that the simulation starts.
  - If you are not allowed to write to the required sub-directory caching will not take place.
  - The cache can be freed per physics system with a button in the panels, or with the CtrlB shortcut key to free it for all selected objects.
  - You may run into trouble if your .blend file path is very long and your operating system has a limit on the path length that is supported.
- Baking:
  - The system is protected against changes after baking.
  - The Bake result is cleared also with CtrlB for all selected objects or click on Free Bake for the current Soft Body system.
  - If the mesh changes the simulation is not calculated anew.

For renderfarms, it is best to bake all the physics systems, and then copy the blendcache to the renderfarm as well.

Interaction in real time

To work with a Soft Body simulation you will find it handy to use the Timeline window. You can change between frames and the simulation will always be shown in the actual state. The option Continue Physics in the Playback menu of the Timeline window lets you interact in real time with the simulation, e.g. by moving collision objects or shake a Soft Body object. And this is real fun!

Continue Physics does not work while playing the animation with AltA

Right. This works only if you start the animation with the Play button of the Timeline window.

You can than select the Soft Body object while running the simulation and Apply the modifier in the Modifiers panel of the Editing context. This makes the deformation permanent.

Tips

- Soft Bodies work especially well if the objects have an even vertex distribution. You need enough vertices for good collisions. You change the deformation (the stiffness) if you add more vertices in a certain region (see the animation of *Image 1b*).
- The calculation of collisions may take a long time. If something is not visible, why calculate it?
- To speed up the collision calculation it is often useful to collide with an additional, simpler, invisible, somewhat larger object (see the example to *Image 1a*).
- Use Soft Bodies only where it makes sense. If you try to cover a body mesh with a tight piece of cloth and animate solely with Soft Body, you will have no success. Self collision of Soft Body hair may be activated, but that is a path that you have to wander alone. We will deal with Collisions in detail later.
- Try and use a Lattice or a Curve Guide Soft Body instead of the object itself. This may be magnitudes faster.

Links

- Developer Notes
- Swinging of a chain
- Softbodies for Rigged Characters

Exterior Forces

Exterior forces are applied to the vertices (and nearly exclusively to the vertices) of Soft Body objects. This is done using Newtons Laws of Physics:

1. If there is no force on a vertex, it stays either unmoved or moves with constant speed in a straight line.
2. The acceleration of a vertex depends on its mass and the force. The heavier the mass of a vertex the slower the acceleration. The larger the force the greater the acceleration.
3. For every action there is an equal and opposite reaction.

Well, this is done only in the range of computing accurateness, there is always a little damping to avoid overshoot of the calculation.

Example

We will begin with a very simple example - the default cube.

- To judge the effect of the external forces you should at first turn off the Goal, so that the vertices are not retracted to their original position.
- Press AltA.

What happens? The cube moves in negative Z-direction. Each of it’s eight vertices is affected by a global, constant force - the gravitation. Gravitation without friction is independent from the weight of an object, so each object you would use as a Soft Body here would fall with the same acceleration. The object does not deform, because every vertex moves with the same speed in the same direction.

Settings

Soft Body Panel

Friction
: The friction of the surrounding medium. The larger the friction, the more viscous is the medium. Friction always appears when a vertex moves relative to it’s surround medium.

Mass
: Mass value for vertices. Larger mass slows down acceleration, except for gravity where the motion is constant regardless of mass. Larger mass means larger inertia, so also braking a Soft Body is more difficult.

Mass Vertex Group
: You can paint weight values for an mesh's mass, and select that vertex group here.

Speed
: You can control the internal timing of the Softbody system with this value. It sets the correlation between frame rate and tempo of the simulation. A free falling body should cover a distance of about five meters after one second. You can adjust the scale of your scene and your simulation with this correlation. If you render with 25 frames per second and 1 meter shall be 1 BU, you have to set Speed to 1.3.

Force Fields

To create other forces you have to use another object, often Empty objects are used for that. You can use some of the forces on Soft Body vertices as on Particles. Soft Bodies react only to:

- Spherical
- Wind
- Vortex

Soft bodies do react on Harmonic fields, but not in a useful way. So if you use a Harmonic field for particles move the Soft body to another layer.

See the section Force Fields for details. The force fields are quite strong, a Spherical field with a strength of -1.0 has the same effect that gravity has - approximately - a force of 10 Newton.

Aerodynamics

This special exterior force is not applied to the vertices but to the connecting edges. Technically, a force perpendicular to the edge is applied. The force scales with the projection of the relative speed on the edge (dot product). Note that the force is the same if wind is blowing or if you drag the edge through the air with the same speed. That means that an edge moving in its own direction feels no force, and an edge moving perpendicular to its own direction feels maximum force.

Simple
: Edges receive a drag force from surrounding media

Lift Force
: Edges receive a lift force when passing through surrounding media.

Factor
: How much aerodynamic force to use. Try a value of 30 at first.

Using a Goal

A goal is a shape that a soft body object tries to conform to.

You have to confine the movement of vertices in certain parts of the mesh, e.g. to attach a Soft Body object at other objects. This is done with the Vertex Group (target). The target position is the original position of the vertex, like it would result from the “normal” animation of an object including Shape Keys, Hooks and Armatures. The vertex tries to reach it’s target position with a certain, adjustable intensity.

Image 2b: Shock absorber description.

Imagine the vertex is connected with it’s target through a spring (*Image 2b*).

Default
: This parameter defines how strong the influence of this spring is. A strength of 1 means, that the vertex will not move as Soft Body at all, instead keep its original position. 0 Goal (or no Goal) means, that the vertex moves only according to Soft Body simulation. If no vertex group is used/assigned, this numeric field is the default goal weight for all vertices. If a vertex group is present and assigned, this button instead shows an popup selector button that allows you to choose the name of the goal vertex group. If you use a vertex group the weight of a vertex defines its goal.
: Often weight painting is used to adjust the weight comfortably. For non-mesh objects the Weight parameter of their vertices/controlpoints is used instead (W in Edit mode, or use the Transform Properties panel). The weight of Hair particles can also be painted in Particle Mode.

Minimum/Maximum
: When you paint the values in vertex-groups (using WeightPaint mode), you can use the G Min and G Max to fine-tune (clamp) the weight values. The lowest vertex-weight (blue) will become G Min, the highest value (red) becomes G Max (please note that the blue-red color scale may be altered by User Preferences).

For now all is applied to single vertices

For now we have discussed vertex movement independent of each other, similar to particles. Every object without Goal would collapse completely if a non uniform force is applied. Now we will move to the next step, the forces that keep the structure of the object and make the Soft Body to a real Body.

Stiffness
: The spring stiffness for Goal. A low value creates very weak springs (more flexible “attachment” to the goal), a high value creates a strong spring (a stiffer “attachment” to the goal).

Dampimg
: The friction of the spring. With a high value the movement will soon come to an end (little jiggle).

Interior Forces

Image 1a: Vertices and forces along their connection edges.

To create a connection between the vertices of a Soft Body object there have to be forces that hold the vertices together. These forces are effective along the edges in a mesh, the connections between the vertices. The forces act like a spring. (*Image 1a*) illustrates how a 3×3 grid of vertices (a mesh plane in Blender) are connected in a Soft Body simulation.

But two vertices could freely rotate if you don't create additional edges between them. Have you ever tried building a storage shelf out of 4 planks alone? Well - don't do it, it will not be stable. The logical method to keep a body from collapsing would be to create additional edges between the vertices. This works pretty well, but would change your mesh topology drastically.

Image 1b: Additional forces with Stiff Quads enabled.

Luckily, Blender allows us to define additional *virtual* connections. On one hand we can define virtual connections between the diagonal edges of a quad face (Stiff Quads, *Image 1b*), on the other hand we can define virtual connections between a vertex and any vertices connected to it's neighbours (Bending Stiffness). In other words, the amount of bend that is allowed between a vertex and any other vertex that is separated by two edge connections.

Edges Settings

The characteristics of edges are set with the Soft Body Edge properties.

Use Edges
: Allow the edges in a Mesh Object to act like springs.

Pull
: The spring stiffness for edges (how much the edges are allowed to stretch). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart. 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly-starched napkin or leather. The Soft Body simulation tends to get unstable if you use a value of 0.999, so you should lower this value a bit if that happens.

Push
: How much the Softbody resist being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

Damp
: The friction for edge springs. High values (max of 50) dampen the Push/Pull effect and calm down the cloth.

Plastic
: Permanent deformation of the object after a collision. The vertices take a new position without applying the modifier.

Bending
: This option creates virtual connections between a vertex and the vertices connected to it's neighbors. This includes diagonal edges. Damping also applies to these connections.

Length
: The edges can shrink or been blown up. This value is given in percent, 0 disables this function. 100% means no change, the body keeps 100% of his size.

Stiff Quads
: For quad faces, the diagonal edges are used as springs. This stops quad faces to collapse completely on collisions (what they would do otherwise).

Shear
: Stiffness of the virtual springs created for quad faces.

Preventing Collapse

To show the effect of the different edge settings we will use two cubes (blue: only quads, red: only tris) and let them fall without any goal onto a plane (how to set up collision is shown on the page Collisions).

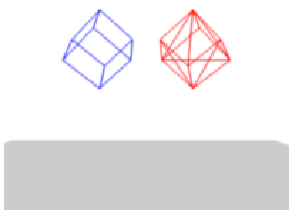


Image 3a: Frame 1 without Stiff Quads.

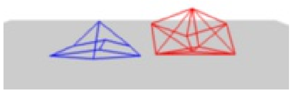


Image 3b: Frame 36.

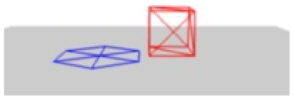


Image 3c: Frame 401.

In (*Image 3*), the default settings are used (without Stiff Quads). The “quad only” cube will collapse completely, the cube composed of tris keeps it’s shape, though it will deform temporarily because of the forces created during collision.

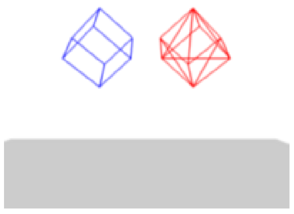


Image 4a: Frame 1 with Stiff Quads.

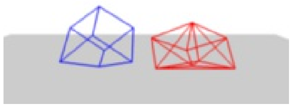


Image 4b: Frame 36.

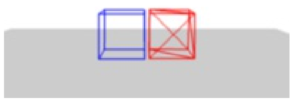


Image 4c: Frame 401.

In (*Image 4*), Stiff Quads is activated (for both cubes). Both cubes keep their shape, there is no difference for the red cube, because it has no quads anyway.

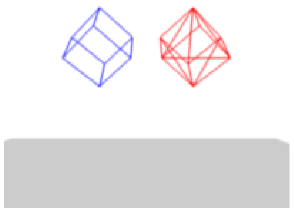


Image 5a: Frame 1 with Bending Stiffness.
[Blend file](#)

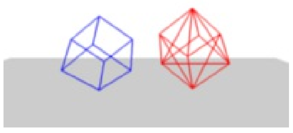


Image 5b: Frame 36.

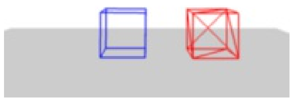


Image 5c: Frame 401.

The second method to stop an object from collapsing is to change it’s Bending Stiffness. This includes the diagonal edges (Damping also applies to these connections).

In (*Image 5*), Be is activated with a strength setting of 1. Now both cubes are more rigid.

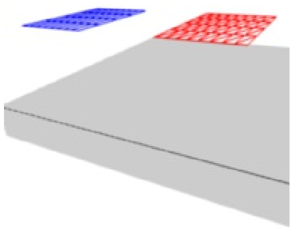


Image 6a: Two planes going to collide.

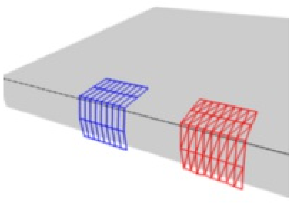


Image 6b: No bending stiffness, Frame 101.

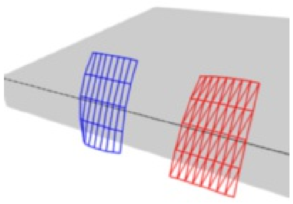


Image 6c: High bending stiffness (10), Frame 101.

Bending stiffness can also be used if you want to make a subdivided plane more plank like. Without Be the faces can freely rotate against each other like hinges (*Image 6b*). There would be no change in the simulation if you activated Stiff Quads, because the faces are not deformed at all in this example.

Bending stiffness on the other hand prevents the plane from being - well - bent.

Collisions

There are two different collision types that you may use: collision between different objects and internal collision. We should set one thing straight from the start: the primary targets of the collision calculation are the vertices of a Soft Body. So if you have too few vertices too few collision takes place. Secondarily, you can use edges and faces to improve the collision calculation.

Collisions with other objects

For a *Soft Body* to collide with another object there are a few prerequisites:

1. Both objects have to share a layer, but the layer does not necessarily have to be visible.
2. The collision object has to be a mesh object.
3. You have to activate the option Collision in the Collision panel of the Physics sub-context (*Image 1*) for the collision object. The collision object may also be a Soft Body.
4. If you use modifiers such as Array and Mirror you have to activate EV.M.Stack to ensure that collision calculation is based on the modified object. The sequence of Modifiers is not important.

Examples

Image 2a: A Soft Body cube colliding with a plane.

Image 2b: A Soft Body plane colliding with a cube - no interaction at all.

Image 2c: Collision with CFace activated.

A cube colliding with a plane works pretty well (*Image 2a*), but a plane falls right through a cube that it is supposed to collide with (*Image 2b*). Why is that? Because the default method of calculation only checks to see if the four vertices of the plane collides with the cube as the plane is pulled down by gravity. You can activate CFace to enable collision between the face of the plane and the object instead (*Image 2c*), but this type of calculation takes much longer.

Let’s have a closer look at the collision calculation, so you can get an idea of how we might optimize it.

Calculating Collisions

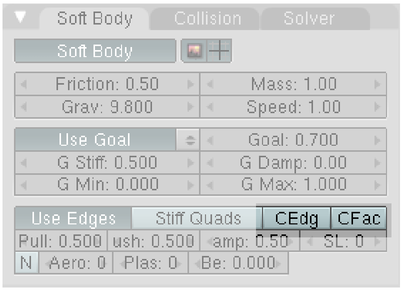
Image 3a: Visualization of the collision of a Soft Body vertex with a plane.

Image 3b: Six Soft Body vertices with different speed. Blend file

Soft Body simulation is by default done on a per vertex basis. If the vertices of the Soft Body do not collide with the collision object there will be no interaction between the two objects.

In (*Image 3a*), you can see a vertex colliding with a plane. If a vertex penetrates the zone between Outer and Inner, it is repulsed by a force in the direction of the face normal. The position that a vertex finally ends up in is dependent on the forces that act upon it. In the example gravity and the repulsion force of the face balance out. The speed at which the vertex is pulled out of the collision zone is influenced by the Choke parameter (*Image 4*).

Now lets see what happens if we make vertices heavier and let them travel at a faster speed. In (*Image 3b*), you can see vertices traveling at different speeds. The two on the far right (5 and 6) are traveling so fast that they pass right through the collision zone (this is because of the default solver precision - which we can fix later). You will notice that the fourth vertex also travels quite fast and because it is heavier it breaches the inner zone. The first three vertices collide OK.

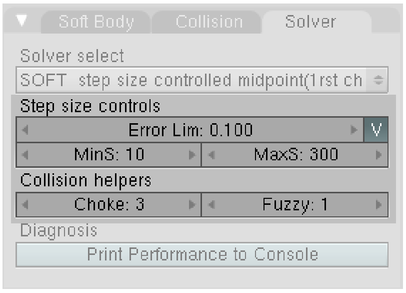


Friction: 0.50 Mass: 1.00
Grav: 9.800 Speed: 1.00
Use Goal Goal: 0.700
G Stiff: 0.500 G Damp: 0.00
G Min: 0.000 G Max: 1.000
Use Edges Stiff Quads CEdg CFac
Pull: 0.500 ush: 0.500 amp: 0.50 SL: 0
N Aero: 0 Plas: 0 Be: 0.000

Image 3d: Also *Edges* and *Faces* can be used for the collision calculation.

You can set up your collision so that edges and even faces are included in the collision calculation (*Image 3d*). The collision is then calculated differently. It is checked whether the edge or face intersects with the collision object, the collision zones are not used.

Good collisions



Solver select
SOFT step size controlled midpoint(1rst ch
Step size controls
Error Lim: 0.100 V
MinS: 10 MaxS: 300
Collision helpers
Choke: 3 Fuzzy: 1
Diagnosis
Print Performance to Console

Image 4: Parameters for Soft Body calculation.

If the collision you have set up is not behaving properly, you can try the following:

The best way

Add Loop Cuts to your Soft Body object in strategic areas that you know are most likely to be involved in a collision.

- The Soft Body object must have more subdivisions than the collision object.
- Check the direction of the face normals.
- If the collision object has sharp spikes they might penetrate the Soft Body.
- The resolution of the solver must match the speed at which Soft Body vertices are traveling. Lower the parameter Error Lim and carefully increase Min S.
- Outer and Inner should be large enough, but zones of opposite faces should not overlap, or you have forces in opposite directions.
- If you use strong forces you should use large zones.
- Set Choke to a high enough value (all the way up if necessary) if you have difficulties with repelled vertices.
- Colliding faces are difficult to control and need long calculation times. Try not to use them.

Often it is better to create a simplified mesh to use as your collision object, however this may be difficult if you are using an animated mesh.

Self Collision

Self Collision is working only if you have activated Use Edges.

When enabled, allows you to control how Blender will prevent the Soft Body from intersecting with itself. Every vertex is surrounded with an elastic virtual ball. Vertices may not penetrate the balls of other vertices. If you want a good result you may have to adjust the size of these balls. Normally it works pretty well with the default options.

Ball Size Caclulation

Man (“manual”)
: The Ball Size directly sets the ball size (in BU).

Av (“average”)
: The average length of all edges attached to the vertex is calculated and then multiplied with the Ball Size setting. Works well with evenly distributed vertices.

Min/Max
: The ball size is as large as the smallest/largest spring length of the vertex multiplied with the Ball Size.

AvMiMax (“average min/max”)
: Size = ((Min + Max)/2) × Ball Size.

Ball Size
: Default 0.49 BU or fraction of the length of attached edges. The edge length is computed based on the algorithm you choose. You know how when someone stands too close to you, and feel uncomfortable? We call that our “personal space”, and this setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self-collision.

Set this value to the fractional distance between vertices that you want them to have their own “space”. Too high of a value will include too many vertices all the time and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there won’t be enough time to slow them down.

Stiffness
: Default 1.0. How elastic that ball of personal space is.

Damping
: Default 0.5. How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

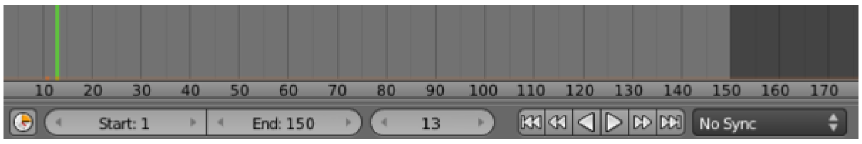
Collisions with other objects are set in the (other) Collision panel. To collide with another object they have to share at least one common layer.

Simple examples

some simple examples showing the power of softbody physics.

bouncing cube

change your start and end frames to 1 and 150.



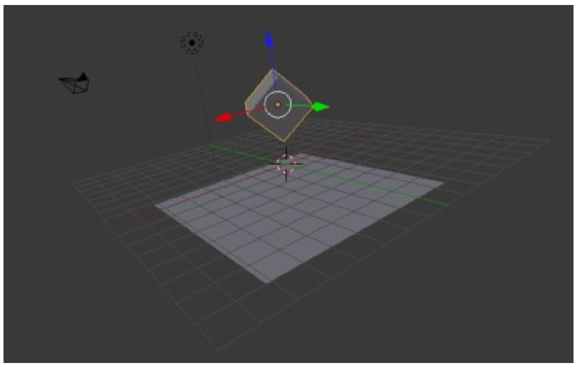
Start: 1 | End: 150 | 13 | No Sync

The timeline

add a plane, and scale it 5 times. next go to the physics tab, and add a collision. the default settings are fine for this example.

now add a cube, or use the default cube. Tab into edit mode and subdivide it thrice. then add a bevel modifier to it, to smoothen the edges. to add a little more, press r twice, and move your cursor a bit.

when finist, your scene should look like this:

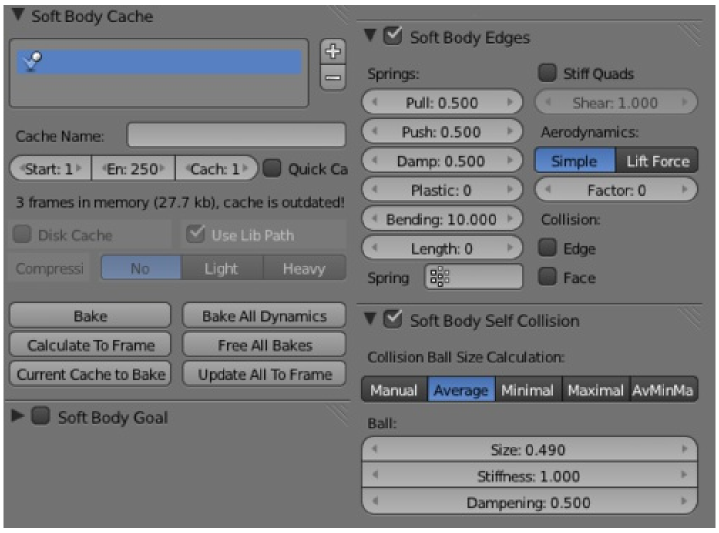

The scene, ready for softbody physics

Everything is ready to add the softbody physics. go to the physics tab and add 'softbody'. uncheck the soft body goal , and check softbody self collision. under soft body edges, increase the bending to 10.

playing tha animation with alt a will now give a slow animation of a bouncing cube. to speed things up, we need to bake the softbody physics.

Under Soft Body Cache change start and end to your start and end frames. in this case 1 and 150. to test if everything is working, you can take a cache step of 5 or 10, but for the final animation it's better to reduce it to 1, to cache everything.

when finist, your physics panel should look like this:



Soft Body Cache

Cache Name:

Start: 1 | En: 250 | Cach: 1 | Quick Ca

3 frames in memory (27.7 kb), cache is outdated!

Disk Cache | ✓ Use Lib Path

Compressi: [✓ No] Light Heavy

Soft Body Goal

✓ Soft Body Edges

Springs: | Stiff Quads
Pull: 0.500 | Shear: 1.000
Push: 0.500 | Aerodynamics:
Damp: 0.500 | [✓ Simple] Lift Force
Plastic: 0 | Factor: 0
Bending: 10.000 | Collision:
Length: 0 | Edge
Spring | Face

✓ Soft Body Self Collision

Collision Ball Size Calculation:

Manual [✓ Average] Minimal Maximal AvMinMa

Ball:
Size: 0.490
Stiffness: 1.000
Dampening: 0.500

The physics settings.

you can now bake the simulation, give the cube materials and textures and render the animation.

result

the rendered bouncing cube:

[video link]

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Soft Body settings

Soft Body
: This creates the soft body modifier on the selected object

Render
: Enable soft body during render

Display
: Display soft body in real time.

Soft Body

Friction
: The friction of the surrounding medium. Generally friction dampens a movement.

Mass
: Mass value for vertices. Larger mass slows down acceleration, except for gravity where the motion is constant regardless of mass. Larger mass means larger inertia, so also braking a Soft Body is more difficult.

Vertex Group Mass
: Use a specified vertex group for mass values

Speed
: You can control the internal timing of the Softbody system with this value.

Soft Body Cache

Start- and Endframe
The Start and End settings in the Collision panel are not only valid for the baking, but for all Soft Body simulations. So if your animation lasts longer than 250 frames, you have to increase the End value.

Cache
: Select cache to use for simulation. Add, and remove caches.

Cache Name
: Specify the name of the cache.

Start/End
: First and last frame of the simulation. Always valid, not only for *baking*.

Cache Step
: Number of frames between cache steps.

Disk Cache
: Save cache files to disk. Blend file must be saved first.

Use Lib Path
: Use this files path when library linked into another file.

Compression
: Compression method to be used

    No
    : No compression.

    Light
    : Fast but not so effective compression.

    Heavy
    : Effective but slow compression.

Bake
: Calculates the simulation and protects the cache. You need to be in Object mode to bake.

Free Bake
: Clears the cache.

Calculate to Frame
: Bake physics to current frame

Current Cache to Bake
: Bake from Cache.

Bake All Dynamics
: Bake all physics

Free All Bakes
: Free all baked caches of all objects in the current scene

Update All To Frame
: Update cache to current frame

If you haven’t saved the blend file the cache is created in memory, so save your file first or the cache may be lost.

Soft Body Goal

Use Goal
: Soft Body Goal acts like a pin on a chosen set of vertices; controlling how much of an effect soft body has on them. Enabling this tells Blender to use the position / animated position of a vertex in the simulation. Animating the vertices can be done in all the usual ways before the Soft Body simulation is applied. The *goal* is the desired end-position for vertices. How a softbody tries to achieve this goal can be defined using stiffness forces and damping.

Default
: If no vertex group is used, this numeric field is the default goal weight for all vertices. If a vertex group is present and assigned, this button instead shows an popup selector button that allows you to choose the name of the goal vertex group. A Goal value of 1.0 means no Soft Body simulation, the vertex stays at its original (animated) position. When setting Goal to 0.0, the object is only influenced by physical laws. By setting goal values between 0.0 and 1.0, you can blend between having the object affected only by the animation system, and having the object affected only by the soft body effect.

Minimum/Maximum
: When you paint the values in vertex-groups (using Weight Paint mode), you can use the G Min and G Max to fine-tune (clamp) the weight values. The lowest vertex-weight (blue) will become G Min, the highest value (red) becomes G Max (please note that the blue-red color scale may be altered by User Preferences).

Stiffness
: The spring stiffness for Goal. A low value creates very weak springs (more flexible “attachment” to the goal), a high value creates a strong spring (a stiffer “attachment” to the goal).

Damping
: The friction for Goal. Higher values dampen the effect of the goal on the soft body.

Vertex Group
: Use a vertex group to specify goal weights.

Soft Body Edges

Use Edges
: The edges in a Mesh Object can act as springs as well, like threads in fabric.

Pull
: The spring stiffness for edges (how much the edges are stretched). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart. 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly-starched napkin or leather.

Push
: How much the Softbody resist being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

Damp
: The friction for edge springs. High values (max of 50) dampen the edge stiffness effect and calm down the cloth.

Plastic
: Plasticity, permanent deformation of the object.

Bending
: This option creates virtual connections between a vertex and the one after the next. This includes diagonal edges. Damping applies also to these connections.

Length
: The edges can shrink or been blown up. This value is given in percent, 0 disables this function. 100% means no change, the body keeps 100% of his size.

Stiff Quads
: For quad faces, the diagonal edges are used as springs. This stops quad faces to collapse completely on collisions (what they would do otherwise).

Shear
: Stiffness of the virtual springs for quad faces.

Aerodynamics
: Simple

    If you turn on Aero the force is not confined to the vertices, but has an effect also on the edges. The angle and the relative speed between medium and edge is used to calculate the force on the edge. This force results that vertices with little connecting edges (front of a plane) fall faster than vertices with more connecting edges (middle of a plane). If all vertices have the same amount of edges in a direction they fall with equal speed. An edge moving in its own direction feels no force, and an edge moving perpendicular to its own direction feels maximum force (think of a straw moving through air). Try it with an Factor of 30 at first.

    Lift Force

    Use an aerodynamic model that is closer to physical laws and looks more interesting. Disable for a more muted simulation.

    Factor

    How much aerodynamic effect to use

Edge
: Checks for edges of the softbody mesh colliding.

Face
: Checks for any portion of the face of the softbody mesh colliding (compute intensive!). While CFace enabled is great, and solves lots of collision errors, there doesn't seem to be any dampening settings for it, so parts of the softbody object near a collision mesh tend to "jitter" as they bounce off and fall back, even when there's no motion of any meshes. Edge collision has dampening, so that can be controlled, but Deflection dampening value on a collision object doesn't seem to affect the face collision.

Soft Body Self Collision

Self Collision is working only if you have activated Use Edges.

Self Collision
: When enabled, allows you to control how Blender will prevent the Soft Body from intersecting with itself. Every vertex is surrounded with an elastic virtual ball. Vertices may not penetrate the balls of other vertices. If you want a good result you may have to adjust the size of these balls. Normally it works pretty well with the default options.

Manual
: The Ball Size directly sets the ball size (in BU).

Averavge ("average")
: The average length of all edges attached to the vertex is calculated and then multiplied with the Ball Size setting. Works well with evenly distributed vertices.

Minimal/Maximal
: The ball size is as large as the smallest/largest spring length of the vertex multiplied with the Ball Size.

AvMiMax
: Size = ((Min + Max)/2) × Ball Size.

Size
: Default 0.49 BU or fraction of the length of attached edges. The edge length is computed based on the algorithm you choose. You know how when someone stands too close to you, and feel uncomfortable? We call that our "personal space", and this setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self-collision.
Set this value to the fractional distance between vertices that you want them to have their own "space". Too high of a value will include too many vertices all the time and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there won't be enough time to slow them down.

Stiffness
: Default 1.0. How elastic that ball of personal space is.

Dampening
: Default 0.5. How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

Collisions with other objects are set in the (other) Collision panel. To collide with another object they have to share at least one common layer.

Soft Body Solver

These settings determine the accurateness of the simulation.

Min Step
: Minimum simulation steps per frame. Increase this value, if the Soft Body misses fast moving collision objects.

Max Step
: Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the Error Limit) but you have probably a good reason to change it.

Auto-Step
: helps the Solver figure out how much work it needs to do based on how fast things are moving.

Error Limit
: Rules the overall quality of the solution delivered. Default 0.1. The most critical setting that says how precise the solver should check for collisions. Start with a value that is 1/2 the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how "bad" it is doing and the Error Limit causes the solver to do some "adaptive step sizing".

Fuzzy
: Fuzziness while on collision, high values make collision handling faster but less stable.

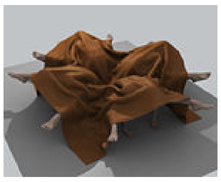
Choke
: Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Print Performance to Console
: Prints on the console how the solver is doing.

Estimate Matrix
: Estimate matrix... split to COM, ROT, SCALE

Cloth Simulation

Cloth example.


Cloth on carved wooden men (made by motorsep).

Cloth example.

Cloth simulation is one of the hardest aspects of CG, because it is a deceptively simple real-world item that is taken for granted, yet actually has very complex internal and environmental interactions. After years of development, Blender has a very robust cloth simulator that is used to make clothing, flags, banners, and so on. Cloth interacts with and is affected by other moving objects, the wind and other forces, as well as a general aerodynamic model, all of which is under your control.

Description

A piece of cloth is any mesh, open or enclosed, that has been designated as cloth. The Cloth panels are located in the Physics sub-context and consist of three panels of options. Cloth is either an open or closed mesh and is mass-less, in that all cloth is assumed to have the same density, or mass per square unit.

Cloth is commonly modeled as a mesh grid primitive, or a cube, but can also be, for example, a teddy bear. However, Blender's Softbody system provides better simulation of closed meshes; Cloth is a specialized simulation of fabrics.

Once the object is designated as Cloth, a Cloth modifier will be added to the object's modifier stack automatically. As a modifier then, it can interact with other modifiers, such as Armature and Smooth. In these cases, the ultimate shape of the mesh is computed in accordance with the order of the modifier stack. For example, you should smooth the cloth *after* the modifier computes the shape of the cloth.

So you edit the Cloth settings in two places: use the F7 Physics buttons to edit the properties of the cloth and use the Modifier stack to edit the Modifier properties related to display and interaction with other modifiers.

You can Apply the cloth modifier to freeze, or lock in, the shape of the mesh at that frame, which removes the modifier. For example, you can drape a flat cloth over a table, let the simulation run, and then apply the modifier. In this sense, you are using the simulator to save yourself a lot of modeling time.

Results of the simulation are saved in a cache, so that the shape of the mesh, once calculated for a frame in an animation, does not have to be recomputed again. If changes to the simulation are made, you have full control over clearing the cache and re-running the simulation. Running the simulation for the first time is fully automatic and no baking or separate step interrupts the workflow.

Computation of the shape of the cloth at every frame is automatic and done in the background; thus you can continue working while the simulation is computed. However it is CPU-intensive and depending on the power of your PC and the complexity of the simulation, the amount of CPU needed to compute the mesh varies, as does the lag you might notice.

Don't jump ahead
If you set up a cloth simulation but Blender has not computed the shapes for the duration of the simulation, and if you jump ahead a lot of frames forward in your animation, the cloth simulator may not be able to compute or show you an accurate mesh shape for that frame, if it has not previously computed the shape for the previous frame(s).

Workflow

A general process for working with cloth is to:

1. Model the cloth object as a general starting shape.
2. Designate the object as a "cloth" in the Physics tab of the Properties window.
3. Model other deflection objects that will interact with the cloth. Ensure the Deflection modifier is last on the modifier stack, after any other mesh deforming modifiers.
4. Light the cloth and assign materials and textures, UV-unwrapping if desired.
5. If desired, give the object particles, such as steam coming off the surface.
6. Run the simulation and adjust Options to obtain satisfactory results. The timeline window's VCR controls are great for this step.
7. Optionally age the mesh to some point in the simulation to obtain a new default starting shape.
8. Make minor edits to the mesh on a frame-by-frame basis to correct minor tears.

Creating Cloth Simulations

This section discusses how to use those options to get the effect you want. First, enable Cloth. Set up for the kind of cloth you are simulating. You can choose one of the presets to have a starting point.

As you can see, the heavier the fabric, the more stiff it is and the less it stretches and is affected by air.

Cloth Panel

Presets
: Contains a number of preset cloth examples, and allows you to add your own.

Quality
: Set the number of simulation steps per frame. Higher values result in better quality, but is slower.

Material

Mass
: The mass of the cloth material.

Structural
: Overall stiffness of the cloth.

Bending
: Wrinkle coefficient. Higher creates more large folds.

Damping

Spring
: Damping of cloth velocity. Higher = more smooth, less jiggling.

Air
: Air normally has some thickness which slows falling things down.

Pinning

Cloth in action.

The first thing you need when pinning cloth is a Vertex Group. There are several ways of doing this including using the Weight Paint tool to paint the areas you want to pin (see the Weight paint section of the manual).

Once you have a vertex group set, things are pretty straightforward; all you have to do is press the Pinning of cloth button in the Cloth panel and select which vertex group you want to use, and the stiffness you want it at.

Stiffness
: Target position stiffness. You can leave the stiffness as it is; the default value of 1 is fine.

Collisions

In most cases, a piece of cloth does not just hang there in 3D space, it collides with other objects in the environment. To ensure proper simulation, there are several items that have to be set up and working together:

1. The Cloth object must be told to participate in Collisions.
2. Optionally (but recommended) tell the cloth to collide with itself.
3. Other objects must be visible to the Cloth object *via* shared layers.
4. The other objects must be mesh objects.
5. The other objects may move or be themselves deformed by other objects (like an armature or shape key).
6. The other mesh objects must be told to deflect the cloth object.
7. The blend file must be saved in a directory so that simulation results can be saved.
8. You then Bake the simulation. The simulator computes the shape of the cloth for a frame range.
9. You can then edit the simulation results, or make adjustments to the cloth mesh, at specific frames.
10. You can make adjustments to the environment or deforming objects, and then re-run the cloth simulation from the current frame forward.

Collision Settings

Cloth Collisions panel.

Now you must tell the Cloth object that you want it to participate in collisions. For the cloth object, locate the Cloth Collision panel, shown to the right:

Enable Collisions
: LMB click this to tell the cloth object that it needs to move out of the way.

Quality
: A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the cloth.

Distance
: As another object gets this close to it (in Blender Units), the simulation will start to push the cloth out of the way.

Repel
: Repulsion force to apply when cloth is close to colliding.

Repel Distance

Maximum distance to apply repulsion force. Must be greater than minimum distance.

Friction
: A coefficient for how slippery the cloth is when it collides with the mesh object. For example, silk has a lower coefficient of friction than cotton.

Self-collisions

Real cloth cannot permeate itself, so you normally want the cloth to self-collide.

Enable Self Collisions
: Click this to tell the cloth object that it should not penetrate itself. This adds to simulation compute time, but provides more realistic results. A flag, viewed from a distance does not need this enabled, but a close-up of a cape or blouse on a character should have this enabled.

Quality
: For higher self-collision quality just increase the Quality and more self collision layers can be solved. Just keep in mind that you need to have at least the same Collision Quality value as the Quality value.

Distance
: If you encounter problems, you could also change the Min Distance value for the self-collisions. The best value is 0.75; for fast things you better take 1.0. The value 0.5 is quite risky (most likely many penetrations) but also gives some speedup.

Regression blend file: Cloth selfcollisions.

Shared Layers

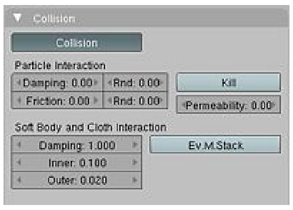
Suppose you have two objects: a pair of Pants on layers 2 and 3, and your Character mesh on layers 1 and 2. You have enabled the Pants as cloth as described above. You must now make the Character “visible” to the Cloth object, so that as your character bends its leg, it will push the cloth. This principle is the same for all simulations; simulations only interact with objects on a shared layer. In this example, both objects share layer 2.

To view/change an object’s layers, RMB click to select the object in Object mode in the 3D view. M to bring up the “Move Layers” popup, which shows you all the layers that the object is on. To put the object on a single layer, LMB click the layer button. To put the object on multiple layers, ⇧ Shift LMB the layer buttons. To remove an object from a selected layer, simply ⇧ Shift LMB the layer button again to toggle it.

Mesh Objects Collide

If your colliding object is not a mesh object, such as a NURBS surface, or text object, you must convert it to a mesh object. To do so, select the object in object mode, and in the 3D View header, select Object → Convert Object Type (AltC), and select Mesh from the popup menu.

Cloth - Object collisions



Damping: 0.00 Rnd: 0.00
Friction: 0.00 Rnd: 0.00
Permeability: 0.00
Damping: 1.000
Inner: 0.100
Outer: 0.020

Collision settings.

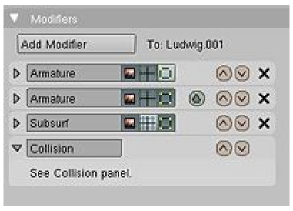
The cloth object needs to be deflected by some other object. To deflect a cloth, the object must be enabled as an object that collides with the cloth object. To enable Cloth - Object collisions, you have to enable deflections on the collision object (not on the cloth object).

In the Buttons window, Object context, Physics sub-context, locate the Collision panel shown to the right. It is also important to note that this collision panel is used to tell all simulations that this object is to participate in colliding/deflecting other objects on a shared layer (particles, soft bodies, and cloth).

Beware

There are three different Collision panels, all found in the Physics sub-context. The first (by default), a tab beside the Fields panel, is the one needed here. The second panel, a tab in the Soft Body group, concern softbodies (and so has nothing to do with cloth). And we have already seen the last one, by default a tab beside the Cloth panel.

Mesh Object Modifier Stack



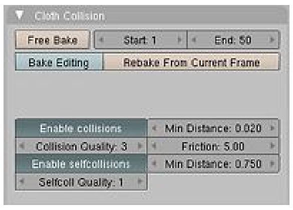
Collision stack.

The object’s shape deforms the cloth, so the cloth simulation must know the “true” shape of that mesh object at that frame. This true shape is the basis shape as modified by shape keys or armatures. Therefore, the Collision modifier must be **after** any of those. The image to the right shows the Modifiers panel for the Character mesh object (not the cloth object).

Cloth Cache

Cache settings for cloth are the same as with other dynamic systems. See Particle Cache for details.

Bake Collision



Start: 1
End: 50
Min Distance: 0.020
Collision Quality: 3
Friction: 5.00
Min Distance: 0.750
Selfcoll Quality: 1

After you have set up the deflection mesh for the frame range you intend to run the simulation (including animating that mesh *via* armatures), you can now tell the cloth simulation to compute (and avoid) collisions. Select the cloth object and in the Object context, Physics sub-context, set the Start and End settings for the simulation frames you wish to compute, and click the Bake button.

You cannot change Start or End without clearing the bake simulation. When the simulation has finished, you will notice you have the option to free the bake, edit the bake and re-bake:

There’s a few things you’ll probably notice right away. First, it will bake significantly slower than before, and it will probably clip through the box pretty badly as in the picture on the right.

Editing the cached simulation=

The cache contains the shape of the mesh at each frame. You can edit the cached simulation, after you've baked the simulation and pressed the Bake Editing button. Just go to the frame you want to fix and ⇆ Tab into Edit mode. There you can move your vertices using all of Blender’s mesh shaping tools. When you exit, the shape of the mesh will be recorded for that frame of the animation. If you want Blender to resume the simulation using the new shape going forward, LMB click ’Rebake from next Frame and play the animation. Blender will then pick up with that shape and resume the simulation.

Edit the mesh to correct minor tears and places where the colliding object has punctured the cloth.

If you add, delete, extrude, or remove vertices in the mesh, Blender will take the new mesh as the starting shape of the mesh back to the *first frame* of the animation, replacing the original shape you started with, up to the frame you were on when you edited the mesh. Therefore, if you change the content of a mesh, when you ⇆ Tab out of Edit mode, you should unprotect and clear the cache so that Blender will make a consistent simulation.

Troubleshooting

If you encounter some problems with collision detection, there are two ways to fix them:

- The fastest solution is to increase the Min Distance setting under the Cloth Collision panel. This will be the fastest way to fix the clipping; however, it will be less accurate and won’t look as good. Using this method tends to make it look like the cloth is resting on air, and gives it a very rounded look.
- A second method is to increase the Quality (in the first Cloth panel). This results in smaller steps for the simulator and therefore to a higher probability that fast-moving collisions get caught. You can also increase the Collision Quality to perform more iterations to get collisions solved.
- If none of the methods help, you can easily edit the cached/baked result in Edit mode afterwards.
- My Cloth is torn by the deforming mesh - he “Hulks Out”: Increase its structural stiffness (StructStiff setting, Cloth panel), very high, like 1000.

Subsurf modifier
A bake/cache is done for every subsurf level so please use **the equal** subsurf level for render and preview.

Examples

To start with cloth, the first thing you need, of course, is some fabric. So, let's delete the default cube and add a plane. I scaled mine up along the Y axis, but you don't have to do this. In order to get some good floppy and flexible fabric, you'll need to subdivide it several times. I did it 8 times for this example. So ⇆ Tab into Edit mode, and press W → Subdivide multi, and set it to 8.

Now, we'll make this cloth by going to the Object context (F7) → Physics sub-context. Scroll down until you see the Cloth panel, and press the Cloth button. Now, a lot of settings will appear, most of which we'll ignore for now.

That's all you need to do to set your cloth up for animating, but if you hit AltA, your lovely fabric will just drop very un-spectacularly. That's what we'll cover in the next two sections about pinning and colliding.

Using Simulation to Shape/Sculpt a Mesh

You can Apply the Cloth modifier at any point to freeze the mesh in position at that frame. You can then re-enable the cloth, setting the start and end frames from which to run the simulation forward.

Another example of aging is a flag. Define the flag as a simple grid shape and pin the edge against the flagpole. Simulate for 50 frames or so, and the flag will drop to its "rest" position. Apply the Cloth modifier. If you want the flag to flap or otherwise move in the scene, re-enable it for the frame range when it is in camera view.

Smoothing of Cloth

Now, if you followed this from the previous section, your cloth is probably looking a little blocky. In order to make it look nice and smooth like the picture you need to apply a Smooth and/or Subsurf modifier in the Modifiers panel under the Editing context (F9). Then, in the same context, find the Links and Materials panel (the same one you used for vertex groups) and press Set Smooth.

Now, if you hit AltA, things are starting to look pretty nice, don't you think?

Cloth on armature

Cloth deformed by armature and also respecting an additional collision object: Regression blend file.

Cloth with animated vertex groups

Cloth with animated pinned vertices: Regression blend file. UNSUPPORTED: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will not work (e.g. from goal = 0 to goal = 0.5).

Cloth with Dynamic Paint

Cloth with Dynamic Paint using animated vertex groups: Regression blend file. UNSUPPORTED: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will not work (e.g. from goal = 0 to goal = 0.5) because the necessary "goal springs" cannot be generated on the fly.

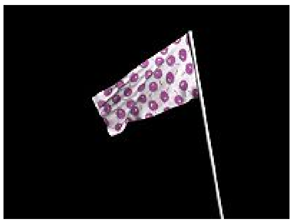
Using Cloth for Softbodies

Using cloth for softbodies.

Cloth can also be used to simulate softbodies. It's for sure not its main purpose but it works nonetheless. The example image uses standard Rubber material, no fancy settings, just AltA.

Blend file for the example image: Using Cloth for softbodies.

Cloth with Wind



Flag with wind applied.

Regression blend file for Cloth with wind and self collisions (also the blend for the image above): Cloth flag with wind and selfcollisions.

Fluid Simulation

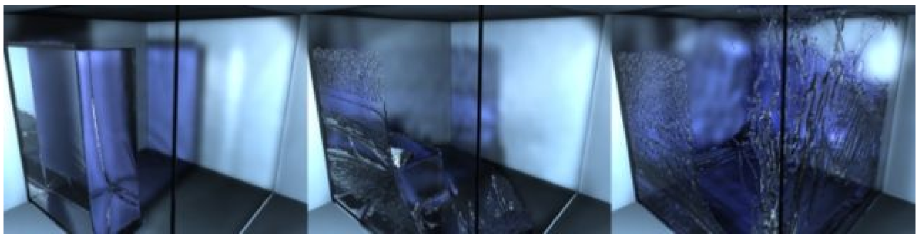
Mode: Object mode / Edit mode (Mesh)

Panel: Physics sub-context → Fluid

Description

While modeling a scene with blender, certain objects can be marked to participate in the fluid simulation, e.g. as fluid or as an obstacle. The bounding box of another object will be used to define a box-shaped region to simulate the fluid in (the so called “simulation domain”). The global simulation parameters (such as viscosity and gravity) can be set for this domain object.

Using the BAKE button, the geometry and settings are exported to the simulator and the fluid simulation is performed, generating a surface mesh together with a preview for each animation frame, and saving them to hard disk. Then the appropriate fluid surface for the current frame is loaded from disk and displayed or rendered.


A breaking dam.

Workflow

In general, you follow these steps:

- set the simulation domain (the portion of the scene where the fluid will flow),
- set the fluid source(s), and specify its material, viscosity, and initial velocity,
- eventually, set other objects to control the volume of the fluid (inlets and outlets),
- eventually, set other objects related to the fluid, like:
  - obstacles,
  - particles floating on the fluid,
  - fluid control, to shape part of the fluid in the desired form,
- eventually, animate the fluid properties,
- Bake the simulation (eventually, revise as necessary and bake repeatedly).

Baking is done on the Domain object!

When you calculate the fluid simulation, **you bake the simulation on the domain object**.

For this reason:

- all the baking options are visible only when selecting the Domain Object,
- baking options are explained in the the baking section of the Domain manual page.

More about the simulation

To know more about simulating fluids in Blender you can read:

- some useful hint about the simulation,
- some technical details, to learn how to do a more realistic fluid simulation,
- the fluids appendix to learn limitations and workarounds, and some additional links.

Fluid Domain

The Domain Object

The bounding box of the object serves as the boundary of the simulation. **All fluid objects must be in the domain.** Fluid objects outside the domain will not bake. No tiny droplets can move outside this domain; it's as if the fluid is contained within the 3D space by invisible force fields. There can be only a single fluid simulation domain object in the scene.

The shape of the object does not matter because it will *always* be treated like a box (The lengths of the bounding box sides can be different). So, usually there won't be any reason to use another shape than a box. If you need obstacles or other boundaries than a box to interfere with the fluid flow, you need to insert additional obstacle objects *inside* the domain boundary.

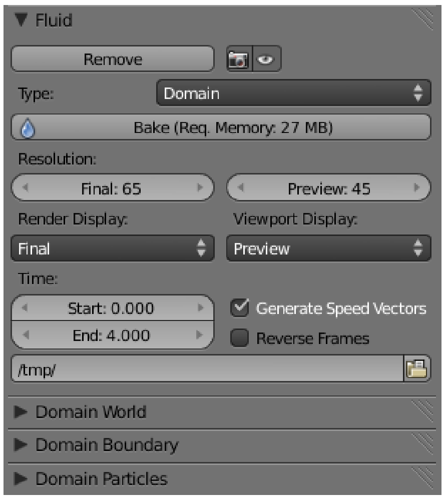
This object will be *replaced* by the fluid during the simulation.

Baking is done on the Domain object

When you calculate the fluid simulation, **you bake the simulation on the domain object**. For this reason all the baking options are visible only when selecting the Domain Object.

For baking options, please refer to the baking section in this page.

Options


▼ Fluid
Remove
Type: Domain
Bake (Req. Memory: 27 MB)
Resolution:
Final: 65
Preview: 45
Render Display: Final
Viewport Display: Preview
Time:
Start: 0.000
End: 4.000
[✓] Generate Speed Vectors
[ ] Reverse Frames
/tmp/
▶ Domain World
▶ Domain Boundary
▶ Domain Particles

The fluid simulation options with Domain selected

Bake button
For baking options please refer to the baking section in this page.

Resolution

Render resolution

The granularity at which the actual fluid simulation is performed. This is probably the most important setting for the simulation as it determines the amount of details in the fluid, the memory and disk usage as well as computational time.

10cm mug at Resolution 70. 10cm mug at Resolution 200.

Note that the amount of required memory quickly increases: a resolution of 32 requires ca. 4MB, 64 requires ca. 30MB, while 128 already needs more than 230MB. Make sure to set the resolution low enough, depending on how much memory you have, to prevent Blender from crashing or freezing. Remember also that many operating systems limit the amount of memory that can be allocated by a single *process*, such as Blender, even if the *machine* contains much more than this. Find out what limitations apply to your machine.

Resolution and Real-size of the Domain
Be sure to set the resolution appropriate to the real-world size of the domain (see the *Realworld-size* in the Domain Wold panel). If the domain is not cubic, the resolution will be taken for the longest side. The resolutions along the other sides will be reduced

according to their lengths (therefore, a non-cubic domain will need less memory than a cubic one, resolutions being the same).

Previewresolution

This is the resolution at which the preview surface meshes will be generated. So it does not influence the actual simulation. Even if “there is nothing to see” in the preview, there might be a thin fluid surface that cannot be resolved in the preview.

Display quality

How to display a baked simulation in the 3d view (menu *Viewport Display*) and for rendering (menu *Render Display*):

- *Geometry*: use the original geometry (before simulation).
- *Preview*: use the preview mesh.
- *Final*: use the final high definition mesh.

When no baked data is found, the original mesh will be displayed by default.

After you have baked a domain, it is displayed (usually) in the Blender window as the preview mesh. To see the size and scope of the original domain box, select Geometry in the left dropdown.

Time

Start

It is the simulation start time (in seconds).

This option makes the simulation computation in Blender start later in the simulation. The domain deformations and fluid flow prior to the start time are not saved.

For example, if you wanted the fluid to appear to already have been flowing for 4 seconds before the actual first frame of data, you would enter 4.0 here.

End

It is the simulation ending time (in seconds).

Start and end times have nothing to do with how many frames are baked

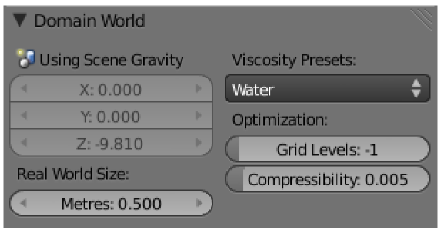
If you set *Start* time to 3.0, and *End* time to 4.0, you will simulate 1 second of fluid motion. That one second of fluid motion will be spread across however-many frames are set in the Anim panel (Scene context → Render sub-context → Anim and Output panel). This means, for example, that if you have Blender set to make 250 frames at 25 fps, the fluid will look like it had already been flowing for 3 seconds at the start of the simulation, *but* will play in slow motion (one-tenth normal speed), since the 1 second fluid sim plays out over the course of 10 video seconds. To correct this, change the end time to 13.0 (3.0 + 10.0) to match the 250 frames at 25 fps. Now, the simulation will be real-time, since you set 10 seconds of fluid motion to simulate over 10 seconds of animation. Having these controls in effect gives you a “speed control” over the simulation.

Generate Speed Vector
If this button is clicked, no speed vectors will be exported. So by default, speed vectors are generated and stored on disk. They can be used to compute image based motion blur with the compositing nodes.

Reverse fluid frames
The simulation is calculated backward

Bake directory
For baking options please refer to the baking section in this page.

Domain World



The Domain World options.

Viscosity
The “thickness” of the fluid and actually the force needed to move an object of a certain surface area through it at a certain speed. You can either enter a value directly or use one of the presets in the drop down (such as honey, oil, or water).
For manual entry, please note that the normal real-world viscosity (the so-called dynamic viscosity) is measured in Pascal-seconds (Pa.s), or in Poise units (P, equal to 0.1 Pa.s, pronounced “*pwaz*”, from the Frenchman Jean-Louis Poiseuille, who discovered the laws on “the laminar flow of viscous fluids”), and commonly centiPoise units (cP, equal to 0.001 Pa.s, “*sentipwaz*”). Blender, on the other hand, uses the kinematic viscosity (which is dynamic viscosity in Pa.s, divided by the density in kg.m^{-3}, unit $m^2.s^{-1}$). The table below gives some examples of fluids together with their dynamic and kinematic viscosities.
Manual entries are specified by a floating point number and an exponent. These floating point and exponent entry fields (scientific notation) simplify entering very small or large numbers. The viscosity of water at room temperature is 1.002 cP, ou

0.001002 Pa.s; the density of water is about 1000 kg.m^{-3}, which gives us a kinematic viscosity of 0.000001002 m^2.s^{-1} - so the entry would be 1.002 times 10 to the minus six (1.002×10^{-6} in scientific notation). Hot Glass and melting iron is a fluid, but very thick; you should enter something like 1.0×10^{0} (= 1.0) as its kinematic viscosity (indicating a value of 1.0×10^{6} cP).
Note that the simulator is not suitable for non-fluids, such as materials that do not "flow". Simply setting the viscosity to very large values will not result in rigid body behavior, but might cause instabilities.

Viscosity varies
The default values in Blender are considered typical for those types of fluids and "look right" when animated. However, actual viscosity of some fluids, especially sugar-laden fluids like chocolate syrup and honey, depend highly on temperature and concentration. Oil viscosity varies by SAE rating. Glass at room temperature is basically a solid, but glass at 1500 degrees Celsius flows (nearly) like water.

Blender Viscosity Unit Conversion

Fluid	dynamic viscosity (in cP)	kinematic viscosity (Blender, in m^2.s^{-1})
Water (20°C)	1.002×10^{0} (1.002)	1.002×10^{-6} (0.000001002)
Oil SAE 50	5.0×10^{2} (500)	5.0×10^{-5} (0.00005)
Honey (20°C)	1.0×10^{4} (10,000)	2.0×10^{-3} (0.002)
Chocolate Syrup	3.0×10^{4} (30,000)	3.0×10^{-3} (0.003)
Ketchup	1.0×10^{5} (100,000)	1.0×10^{-1} (0.1)
Melting Glass	1.0×10^{15}	1.0×10^{0} (1.0)

Realworld-size
Size of the domain object in the real world in meters. If you want to create a mug of coffee, this might be 10 cm (0.1 meters), while a swimming pool might be 10m. The size set here is for the longest side of the domain bounding box.

Optimization

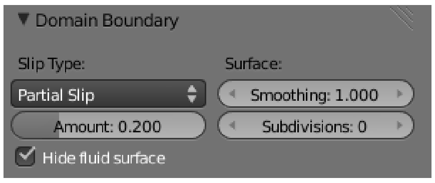
Gridlevel

How many adaptive grid levels to be used during simulation - setting this to -1 will perform automatic selection.

Compressibility

If you have problems with large standing fluid regions at high resolution, it might help to reduce this number (note that this will increase computation times).

Domain Boundary



The Domain Boundary panel

This box has all the slip and surface options.

Boundary type
Determines the stickiness of the obstacle surface, called "Surface Adhesion". Surface Adhesion depends in real-world on the fluid and the graininess or friction/adhesion/absorption qualities of the surface:

No Slip

Causes the fluid to stick to the obstacle (zero velocity).

Free Slip

Allows movement along the obstacle (only zero normal velocity).

Part Slip

Mixes both types, with 0 being equal as *No Slip*, and 1 being identical to *Free Slip*.

Surface

Surface Smoothing

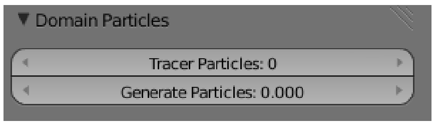
Amount of smoothing to be applied to the fluid surface. 1.0 is standard, 0 is off, while larger values increase the amount of smoothing.

Subdivisions

Allows the creation of high-res surface meshes directly during the simulation (as opposed to doing it afterwards like a subdivision modifier). A value of 1 means no subdivision, and each increase results in one further subdivision of each fluid voxel. The resulting meshes thus quickly become large, and can require large amounts of disk space. Be careful in combination with large smoothing values - this can lead to long computation times due to the surface mesh generation.

Hide fluid surface

Domain Particles


▼ Domain Particles

Tracer Particles: 0

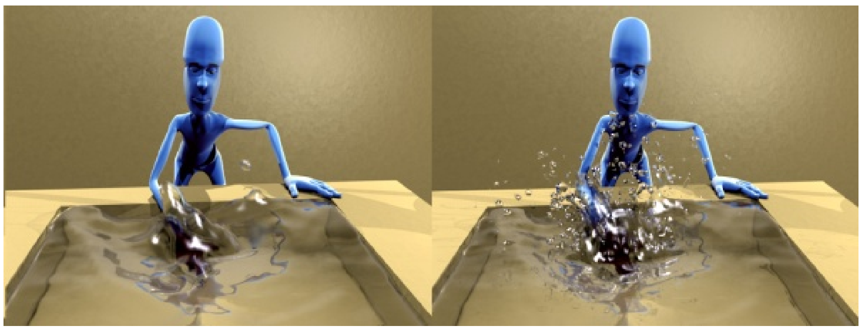
Generate Particles: 0.000

The Domain Particles panel

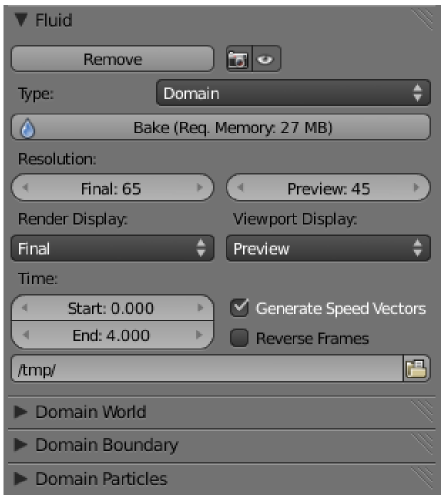
Here you can add particles to the fluid simulated, to enhance the visual effect.

Tracer Particles
: Number of tracer particles to be put into the fluid at the beginning of the simulation. To display them create another object with the Particle fluid type, explained below, that uses the same bake directory as the domain.

Generate Particles
: Controls the amount of fluid particles to create (0=off, 1=normal, >1=more). To use it, you have to have a surface subdivision value of at least 2.


An example of the effect of particles can be seen here - the image to the left was simulated without, and the right one with particles and subdivision enabled.

Baking


▼ Fluid

Remove

Type: Domain

Bake (Req. Memory: 27 MB)

Resolution:

Final: 65 | Preview: 45

Render Display: Final

Viewport Display: Preview

Time:

Start: 0.000

End: 4.000

[✓] Generate Speed Vectors

[ ] Reverse Frames

/tmp/

▶ Domain World

▶ Domain Boundary

▶ Domain Particles

The fluid simulation options with Domain selected

Bake Button

Perform the actual fluid simulation. Blender will continue to work normally, except there will be a status bar in the top of the window, next to the render pulldown. Pressing Esc or the "x" next to the status bar will abort the simulation. Afterwards two "`.bobj.gz`" (one for the Final quality, one for the Preview quality), plus one "`.bvel.gz`" (for the Final quality) will be in the selected output directory for each frame.

Bake directory

REQUIRED!

Directory and file prefix to store baked surface meshes.

This is similar to the animation output settings, only selecting a file is a bit special: when you select any of the previously generated surface meshes (e.g. "`test1_fluidsurface_final_0132.bobj.gz`"), the prefix will be automatically set ("`test1_`" in this example). This way the simulation can be done several times with different settings, and allows quick changes between the different sets of surface data.

The default value is “`/tmp/`”, which is probably *not* what you want. Choose an appropriate directory-name and file prefix so that these files will be stored in an appropriate location *and* named in such a way that two different fluid-simulations won’t conflict with one another (if you’re intending to specify only a directory-name here, i.e. without a filename-prefix, don’t forget the trailing “`/`”).

Notes

Unique domain
Because of the possibility of spanning and linking between scenes, there can only be one domain in an entire .blend file.

Selecting a Baked Domain
After a domain has been baked, it changes to the fluid mesh. To re-select the domain so that you can bake it again after you have made changes, go to any frame and select (RMB) the fluid mesh. Then you can click the BAKE button again to recompute the fluid flows inside that domain.

Baking always starts at Frame #1:
The fluid simulator disregards the Sta setting in the Anim panel, it will always bake from frame 1.
If you wish the simulation to start later than frame 1, you must key the fluid objects in your domain to be inactive until the frame you desire to start the simulation. See below for more information.

Baking always ends at the End Frame set in the Anim panel:
If your frame-rate is 25 frames per second, and ending time is 4.0 seconds, then you should (if your start time is 0) set your animation to end at frame `4.0 × 25 =` 100.

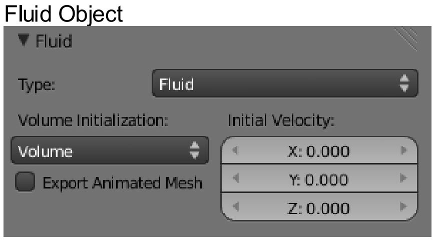
Freeing the previous baked solutions
Deleting the content of the “Bake” directory is a destructive way to achieve this. Be careful if more than one simulation uses the same bake directory (be sure they use different filenames, or they will overwrite one another)!

Reusing Bakes
Manually entering (or searching for) a previously saved (baked) computational directory and filename mask will switch the fluid flow and mesh deformation to use that which existed during the old bake. Thus, you can re-use baked flows by simply pointing to them in this field.

Baking processing time
Baking takes a **lot** of compute power (hence time). Depending on the scene, it might be preferable to bake overnight.

If the mesh has modifiers, the rendering settings are used for exporting the mesh to the fluid solver. Depending on the setting, calculation times and memory use might exponentially increase. For example, when using a moving mesh with Subsurf as an obstacle, it might help to decrease simulation time by switching it off, or to a low subdivision level. When the setup/rig is correct, you can always increase settings to yield a more realistic result.

Fluid Object


▼ Fluid

Type: Fluid

Volume Initialization: Volume

Export Animated Mesh

Initial Velocity:

X: 0.000

Y: 0.000

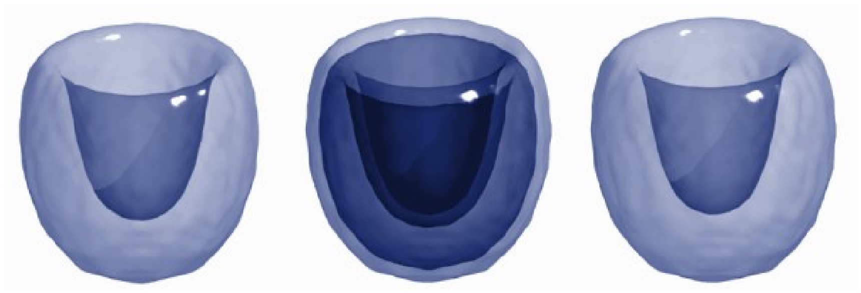
Z: 0.000

Fluid object settings

All regions of this object that are inside the domain bounding box will be used as actual fluid in the simulation. If you place more than one fluid object inside the domain, they should currently not intersect. Also make sure the surface normals are pointing outwards. In contrast to domain objects, the actual mesh geometry is used for fluid objects.

Volume initialization type

- Volume will initialize the inner part of the object as fluid. This works only for closed objects.
- Shell will initialize only a thin layer for all faces of the mesh. This also works for non closed meshes.
- Both combines volume and shell - the mesh should also be closed. See the picture below.


Example of the different volume init types: Volume, Shell and Both (the shell is usually slightly larger than the inner volume)

Animated Mesh/Export
: Click this button if the mesh is animated (e.g. deformed by an armature, shape keys or a lattice). Note that this can be significantly slower, and is not required if the mesh is animated with position or rotation Ipos (i.e. only *object* transformations).

Initial velocity
: Speed of the fluid at the beginning of the simulation, in meters per second.

The direction of Surface Normals makes a big difference!

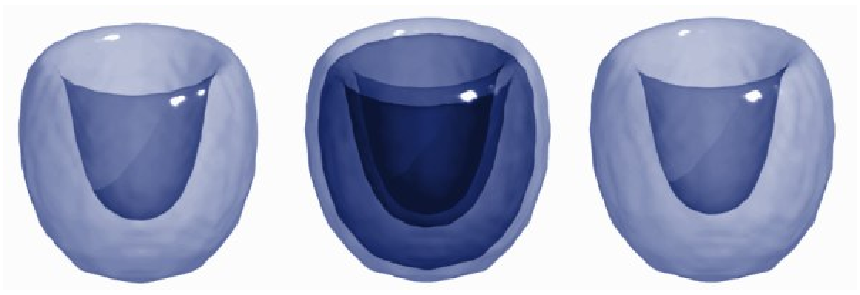
Blender uses the orientation of the Surface Normals to determine what is “inside of” the Fluid object and what is “outside”. You want all of the normals to face *outside* (in Edit mode, use CtrlN or press Space and choose Edit → Normals → Calculate Outside). If the normals face the wrong way, you’ll be rewarded with a “gigantic flood of water” because Blender will think that the volume of the object is outside of its mesh! This applies regardless of the Volume init type setting.

Fluid Obstacle

This object will be used as an obstacle in the simulation. As with a fluid object, obstacle objects currently should not intersect. As for fluid objects, the actual mesh geometry is used for obstacles. For objects with a volume, make sure that the normals of the obstacle are calculated correctly, and radiating properly (use the Flip Normal button, in Edit mode, Mesh Tools panel, Editing context [F9]), particularly when using a spinned container. Applying the Modifier SubSurf before baking the simulation could also be a good idea if the mesh is not animated.

Volume initialization type

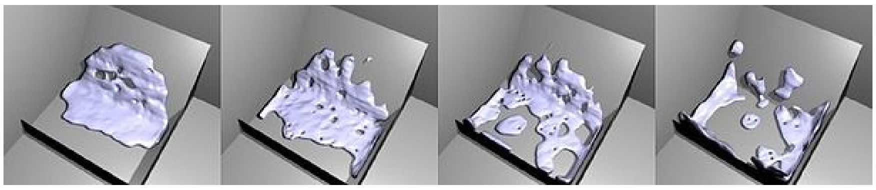
- Volume will initialize the inner part of the object as fluid. This works only for closed objects.
- Shell will initialize only a thin layer for all faces of the mesh. This also works for non closed meshes.
- Both combines volume and shell - the mesh should also be closed. See the picture below.


Example of the different volume init types: Volume, Shell and Both (the shell is usually slightly larger than the inner volume)

Boundary type
Determines the stickiness of the obstacle surface, called “Surface Adhesion”. Surface Adhesion depends in real-world on the fluid and the graininess or friction/adhesion/absorption qualities of the surface.

- Noslip causes the fluid to stick to the obstacle (zero velocity).
- Free(-slip) allows movement along the obstacle (only zero normal velocity).
- Part(-slip) mixes both types, with 0 being mostly noslip, and 1 being identical to freeslip.

Note that if the mesh is moving, it will be treated as noslip automatically.


Example of the different boundary types for a drop falling onto the slanted wall. From left to right: no-slip, part-slip 0.3, part-slip 0.7 and free-slip.

Animated Mesh/Export
Click this button if the mesh is animated (e.g. deformed by an armature, shape keys or a lattice). Note that this can be significantly slower, and is not required if the mesh is animated with position or rotation Ipos (i.e. only *object* transformations).

PartSlip Amount
Amount of mixing between no- and free-slip, described above.

Moving obstacles support

Blender supports now moving obstacles.

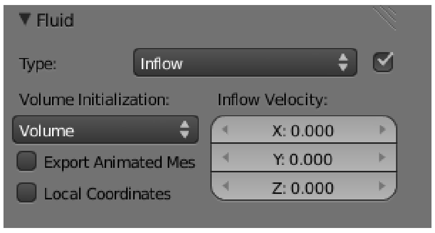
In the past, a moving obstacle was automatically treated as no slip (sticky), so if you wanted to splash off of a moving object, you had to put a transparent plane in the spot where the fluid will hit the moving object, exactly aligned and shaped as the object, to fake the splash. This is not needed anymore.

Impact Factor
Amount of fluid volume correction for gain/loss from impacting with moving objects. If this object is not moving, this setting has no effect. However, it if is and the fluid collides with it, a negative value takes volume away from the Domain, and a positive number adds to it. Ranges from -2.0 to 10.0.

Controlling the fluid volume

To control the volume of the fluid simulation, you can set objects in the scene to add or absorb fluid within the Fluid Domain.

Inflow


▼ Fluid

Type: Inflow ✓

Volume Initialization: Volume

Export Animated Mes

Local Coordinates

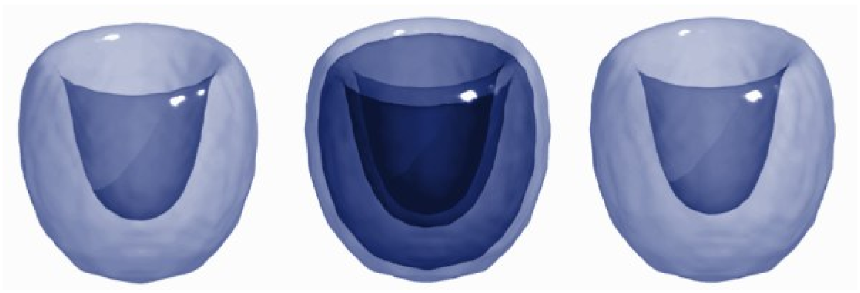
Inflow Velocity: X: 0.000 Y: 0.000 Z: 0.000

Fluid inflow settings

This object will put fluid into the simulation, like a water tap.

Volume initialization type

- Volume will initialize the inner part of the object as fluid. This works only for closed objects.
- Shell will initialize only a thin layer for all faces of the mesh. This also works for non closed meshes.
- Both combines volume and shell - the mesh should also be closed. See the picture below.

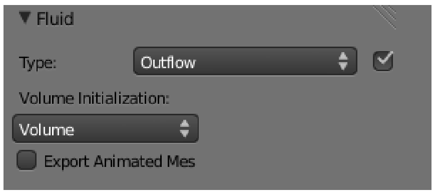

Example of the different volume init types: Volume, Shell and Both (the shell is usually slightly larger than the inner volume)

Inflow velocity
: Speed of the fluid that is created inside of the object.

Local Coords/Enable
: Use local coordinates for the inflow. This is useful if the inflow object is moving or rotating, as the inflow stream will follow/copy that motion. If disabled, the inflow location and direction do not change.

Animated Mesh/Export
: Click this button if the mesh is animated (e.g. deformed by an armature, shape keys or a lattice). Note that this can be significantly slower, and is not required if the mesh is animated with position or rotation Ipos (i.e. only *object* transformations).

Outflow


▼ Fluid

Type: Outflow ✓

Volume Initialization: Volume

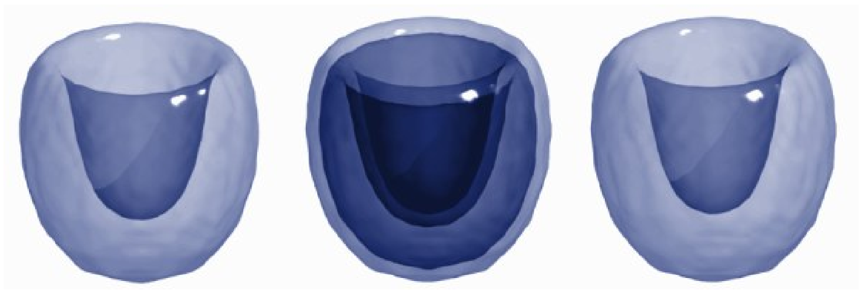
Export Animated Mes

Fluid outflow settings

Any fluid that enters the region of this object will be deleted (think of a drain or a black hole). This can be useful in combination with an inflow to prevent the whole domain from filling up. When enabled, this is like a tornado (waterspout) or “wet vac” vacuum cleaner, and the part where the fluid disappears will follow the object as it moves around.

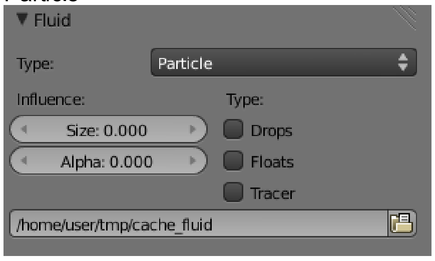
Volume initialization type

- Volume will initialize the inner part of the object as fluid. This works only for closed objects.
- Shell will initialize only a thin layer for all faces of the mesh. This also works for non closed meshes.
- Both combines volume and shell - the mesh should also be closed. See the picture below.


Example of the different volume init types: Volume, Shell and Both (the shell is usually slightly larger than the inner volume)

Animated Mesh/Export
: Click this button if the mesh is animated (e.g. deformed by an armature, shape keys or a lattice). Note that this can be significantly slower, and is not required if the mesh is animated with position or rotation Ipos (i.e. only *object* transformations).

Particle



Fluid particle settings

This type can be used to display particles created during the simulation. For now only tracers swimming along with the fluid are supported. Note that the object can have any shape, position or type - once the particle button is pressed, a particle system with the fluid simulation particles will be created for it at the correct position. When moving the original object, it might be necessary to delete the particle system, disable the fluidsim particles, and enable them again. The fluidsim particles are currently also unaffected by any other particle forces or settings.

Influence
: Size Influence

    The particles can have different sizes, if this value is 0 all are forced to be the same size.

: Alpha Influence

    If this value is >0, the alpha values of the particles are changed according to their size.

Particle type
: Drops

    Surface splashes of the fluid result in droplets being strewn about, like fresh water, with low Surface Tension.

: Floats

    The surface tension of the fluid is higher and the fluid heavier, like cold seawater and soup. Breakaways are clumpier and fall back to the surface faster than Drops, as with high Surface Tension.

: Tracer

    Droplets follow the surface of the water where it existed, like a fog suspended above previous fluid levels. Use this to see where the fluid level has been.

Path (bake directory)
: The simulation run from which to load the particles. This should usually have the same value as the fluid domain object (e.g. copy by CtrlC, CtrlV).

Control

Description

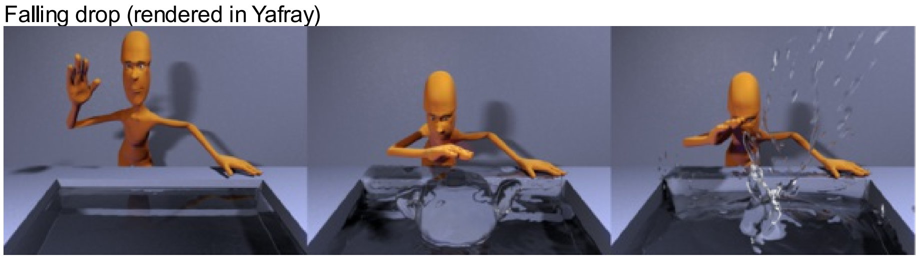
Using the Lattice-boltzman method, the fluid is controlled using particles which define local force fields and are generated automatically from either a physical simulation or a sequence of target shapes. At the same time, as much as possible of the natural fluid motion is preserved.

[video link]

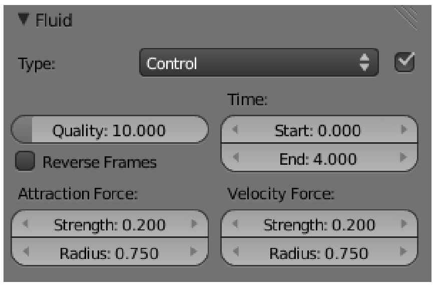
Examples

In this examples, we use the Fluid Control option to control part of the fluid so that it has a certain shape (the sphere drop or the teapot drop) before it falls in the rest of the fluid:

Falling drop (rendered in Yafray)


"Magic Fluid Control"

Options



Type: Control ✓

Time:

Quality: 10.000

Start: 0.000

End: 4.000

Reverse Frames

Attraction Force:

Strength: 0.200

Radius: 0.750

Velocity Force:

Strength: 0.200

Radius: 0.750

Fluid control options.

Quality
: Higher quality result in more control particles for the fluid control object.

Reverse Frames
: The control particle movement gets reversed.

Time
: You specify the start and end time during which time the fluid control object is active.

Attraction force
: The attraction force specifies the force which gets emitted by the fluid control object. Positive force results in attraction of the fluid, negative force in avoidance.

Velocity force
: If the fluid control object moves, the resulting velocity can also introduce a force to the fluid.

See also

Release notes: Template:Release_Notes/2.48/FluidControl

Animating the fluid properties

A new type of Ipo Curve, FluidSim, is available for fluid domain objects. Unlike most other animatable values in Blender, FluidSim Ipos cannot be keyframed by simply using the I key; you must manually set values by clicking in the Ipo window. In order to set a keyframe, you must select the property you wish to animate in the Ipo window and Ctrl LMB click to set the keyframe to the desired location in the Ipo window.

Enter Properties

Note that you do not have to be exact on where you click; we recommend that after you set the control point, open the Transform Properties panel (N) and round the X value to a whole frame number, and then set the Y value that you wish.

The fluid domain has several channels that control the fluid over time:

Fac-Visc
: A multiplicative factor in the fluid's viscosity coefficient. It must be set before baking, and changes the viscosity of the fluid over time, so you can turn water into wi… oil, for example!

Fac-Tim
: Changes the speed of the simulation; like the Speed Control in the VSE can speed up or slow down a video, this curve can speed up or slow down the fluid motion during a frame sequence. If the value for Fac-Tim is less than or equal to zero, time (and the fluid) stands still; the fluid freezes. For values between 0.0 and 1.0, the fluid runs slower and will look thicker. 1.0 is normal fluid motion, and values greater than 1.0 will make the fluid flow faster, possibly appearing thinner.

GravX/GravY/GravZ
: The XYZ vector for gravity changes; aka inertia of the fluid itself (think drinking a cup of coffee while driving NASCAR at Talladega, or sipping an espresso on the autobahn, or watering the plants on the Space Shuttle). Changes in these curves make the fluid slosh around due to external forces.

The Fluid, Obstacle, Inflow, Outflow and Particle objects can use the following channels:

VelX/VelY/VelZ
: Spurts of water from the garden hose can be simulated via these curves, to mimic changes in pressure and/or direction. It also can be used to simulate the effect of wind on a stream of water, for example.

Active
: When Active transitions from 0.0 to something greater than 0 (such as between 0.1 and 1.0), the object's function (designated as an Inflow, or Outflow, etc.) resumes its effect. Crossing down to 0.0 and then at some point, back up, re-establishes the effect and the resulting fluid sim. Use this for dripping, or any kind of intermittent inflow. This active status also works for objects designated as Outflow and Obstacle, so you can also simulate (for example) a drain plugging up.

You can also control the force settings of Control objects:

AttrForceStr, AttrForceRa
: These curves control the values of the attraction force settings.

VelForceStr, VelForceRa
: These curves control the values of the velocity force settings.

Fluid Hints

Some useful hints about fluid simulation in Blender:

- Don’t be surprised, but you’ll get whole bunch of mesh (.bobj.gz) files after a simulation. One set for preview, and another for final. Each set has a .gz file for each frame of the animation. Each file contains the simulation result - so you’ll need them.
- Currently these files will not be automatically deleted, so it is a good idea to e.g. create a dedicated directory to keep simulation results. Doing a fluid simulation is similar to clicking the ANIM button - you currently have to take care of organizing the fluid surface meshes in some directory yourself. If you want to stop using the fluid simulation, you can simply delete all the `*fluid*.bobj.gz` files.
- Before running a high resolution simulation that might take hours, check the overall timing first by doing lower resolution runs.
- Fluid objects must be completely inside the bounding box of the domain object. If not, baking may not work correctly or at all. Fluid and obstacle objects can be meshes with complex geometries. Very thin objects might not appear in the simulation, if the chosen resolution is too coarse to resolve them (increasing it might solve this problem).
- Note that fluid simulation parameters, such as inflow velocity or the active flag can be animated with Fluidsim Ipos (see above).
- Don’t try to do a complicated scene all at once. Blender has a powerful compositor that you can use to combine multiple animations.

  For example, to produce an animation showing two separate fluid flows while keeping your domain small, render one .avi using the one flow. Then move the domain and render another .avi with the other flow using an alpha channel (in a separate B&W .avi?). Then, composite both .avi’s using the compositor’s add function. A third .avi is usually the smoke and mist and it is laid on top of everything as well. Add a rain sheet on top of the mist and spray and you’ll have quite a storm brewing! And then lightning flashes, trash blowing around, all as separate animations, compositing the total for a truly spectacular result.
- If you’re having trouble, or something isn’t working as you think it should - let me know: send the .blend file and a problem description to `nils at thuerey dot de`. Please check these wiki pages and the blenderartists-forum before sending a mail!

Fluid Technical Details

Physical correctness

“My cup runneth over”, created with Blender and Yafray.

Fluid animation can take a lot of time - the better you understand how it works, the easier it will be to estimate how the results will look. The algorithm used for Blender’s fluid simulation is the *Lattice Boltzmann Method* (LBM); other fluid algorithms include *Navier-Stokes* (NS) solvers and *Smoothed Particle Hydrodynamics* (SPH) methods. LBM lies somewhere between these two.

In general, it is really hard for current computers to correctly simulate even a 1-meter tank of water. For simulating a wave crashing through a city, you would probably need one of the most expensive supercomputers you could get, and it might still not work properly, no matter which of the three algorithms above you’re using. Therefore, to achieve “the effect that you really want”, you’ll need to resort to strategies very similar to what filmmakers have been doing (quite successfully…) in “analogue” movies for many years: “*fake it!*”

A good fluid simulation is a *very important* part, but not the *only* part, of achieving a satisfactory image. Let Blender do the computational dirty-work of calculating the basic fluid simulation, then create realism by adding carefully selected details that match the viewer’s expectations for “the real-life maelstrom that you have created”.

For example, you can pretend to have a wave in a gigantic city by: building a *smaller* model, modeling a *small* wave in the model at very high resolution, and hope that nobody will notice the difference between a 100m and a 1m wave (they won’t). Texture the wave front with lots of noise and clouds affecting the color. Add lots of smoke (mist) emitters on the various surfaces that the wave hits, timing each of them to emit at the moment of impact in a direction incident to the surface and collision. Animate cars and trash (and drowning people…) to float and bob on the wave front using the baked mesh. Use a string of mist emitters pointing up positioned at the wave crest to simulate the mist that blows off the top of the crest into the air. Consider exactly where you want to put the camera, whether you want to use a zoom lens or a wide angle, and so on (is the viewer to be “looking down upon the poor unfortunate actors”, or “drowning along with them”?). *This* is the kind of attention to detail, above and beyond the fluid simulation itself, that will carry the shot.

For Blender’s LBM solver, the following things will make the simulation harder to compute:

- Large domains.
- Long duration.
- Low viscosities.
- High velocities.

The viscosity of water is already really low, so especially for small resolutions, the turbulence of water can not be correctly captured. If you look closely, most simulations of fluids in computer graphics do not yet look like real water as of now. Generally, don’t rely on the physical settings too much (such as physical domain size or length of the animation in seconds). Rather try to get the overall motion right with a low resolution, and then increase the resolution as much as possible or desired.

Acknowledgements

The integration of the fluid simulator was done as a Google Summer-of-Code project. More information about the solver can be found at www.ntoken.com. These Animations were created with the solver before its integration into blender: Adaptive Grids, Interactive Animations. Thanks to Chris Want for organizing the Blender-SoC projects, and to Jonathan Merrit for mentoring this one! And of course thanks to Google for starting the whole thing… SoC progress updates were posted here: SoC-Blenderfluid-Blog at PlanetSoC.

The solver itself was developed by Nils Thuerey with the help and supervision of the following people: U. Ruede, T. Pohl, C. Koerner, M. Thies, M. Oechsner and T. Hofmann at the Department of Computer Science 10 (System Simulation, LSS) in Erlangen, Germany.

http://www10.informatik.uni-erlangen.de/~sinithue/img/lsslogo.png

http://www10.informatik.uni-erlangen.de/~sinithue/img/unierlangenlogo.png

Fluid Appendix

Limitations & Workarounds

- One domain per blender file (as of Version 2.42), but you can have multiple fluid objects.

  Workaround: For previews, move the domain around to encompass each fluid flow part, and then for final, scale up the size of the domain to include all fluid objects (but computation will take longer). This is actually a benefit, because it lets you control how much compute time is used, by varying the size and location of the domain.
- If the setup seems to go wrong, make sure all the normals are correct (hence, enter Edit mode, select all, and recalculate normals once in a while).
- Currently there's a problem with zero gravity simulation - simply select a very small gravity until this is fixed.
- If an object is initialized as Volume, it has to be closed and have an inner side (a plane won't work). To use planes, switch to Shell, or extrude the plane.
- Blender freezes after clicking BAKE. Pressing Esc makes it work again after a while - this can happen if the resolution is too high and memory is swapped to hard disk, making everything horribly slow. Reducing the resolution should help in this case.
- Blender crashes after clicking BAKE - this can happen if the resolution is really high and more than 2GB are allocated, causing Blender to crash. Reduce the resolution. Many operating systems limit the total amount of memory that can be allocated by a *process*, such as Blender, even if the *machine* has more memory installed. Sux…
- The meshes should be closed, so if some parts of e.g. a fluid object are not initialized as fluid in the simulation, check that all parts of connected vertices are closed meshes. Unfortunately, the Suzanne (monkey) mesh in Blender is not a closed mesh (the eyes are separate).
- If the fluid simulation exits with an error message (stating e.g. that the "init has failed"), make sure you have valid settings for the domain object, e.g. by resetting them to the defaults.
- To import a single fluid surface mesh you can use this script: .bobj.-Import-Script.
- You may not be able to bake a fluid that takes more than 1GB, not even with the LargeAddressAware build - it might be a limitation of the current fluid engine.
- Note that first frame may well take only a few hundred MBs of RAM memory, but latter ones go over one GB, which may be why your bake fails after awhile. If so, try to bake one frame at the middle or end at full res so you'll see if it works.
- Memory used doubles when you set surface subdivision from 1 to 2.
- Using "generate particles" will also add memory requirements, as they increase surface area and complexity. Ordinary fluid-sim generated particles probably eat less memory.

See also

to do check these links, make sure they are compatible with Blender 2.6

- Tutorial 1: Very Basic Introduction
- Tutorial 2: The Next Step
- Tutorial 1&2 Gui Changes for newer builds
- Another BSoD fluid tutorial
- Developer documentation (implementation, dependencies, …)

External links

to do check these links, make sure they are compatible with Blender 2.6

- An Introduction to Fluid Simulations in Blender (video) (Blendernation link)

  Learn the basics of how to set up a fluid simulation in Blender with an obstacle.
- Fluid Simulator Tutorial (video) (Blendernation link)

  Very easy to understand video-tutorial to fluid simulation newcomers. Also covers some of the most common pitfalls.
- Guide on Blender Fluid Simulator's Parameters (Blendernation link)

Smoke Simulation

Development notes

Blender's new smoke simulation is based on the paper 'Wavelet Turbulence for Fluid Simulation' and associated sample code.

It has been implemented in Blender by Daniel Genrich and Miika Hamalainen.

Inner working

The simulator uses a volumetric fluid-based model, with the end results output as voxel grids. This voxel data is visualized interactively in Blender's 3D view using custom OpenGL shading, and can be rendered using the Voxel Data texture. Blender's **smoke simulation** wraps Voxels around existing Particles. It requires a particle-emitting object and a 'domain' object within which smoke is rendered.

Note
This Part of the Documentation uses the 2.58 Release

User workflow

The smoke simulation is similar to the Fluid simulation: a Domain and Flow object is required to do a smoke simulation:

- set as the simulation Domain an object that defines the bounds of the simulation volume,
- set as the Flow object an object which determines where the smoke will be produced from,
- set Collision objects, to make the smoke interact with objects in the scene.
- assign a Material to the smoke
- save the project
- bake the simulation

In case you are having troubles, please consult the Appendix

Smoke Domain

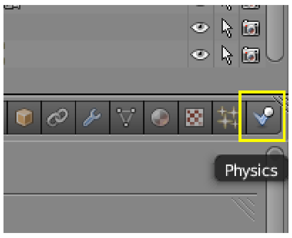
Like the Fluid Sim, most of the settings are found when the Domain Object is selected.

Creating the Domain

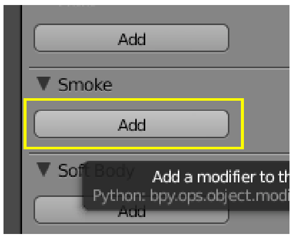
Before you can add smoke to your scene you need to define the area where the smoke simulation takes place. In Blender physics this is called a domain. A good idea is to choose a cube for that because you can scale it to the view of your camera later on. In our case we just make the default cube bigger by hitting S and dragging the mouse.

Don't edit the domain's vertices!
If you want a bigger domain, scale the object. Changing it in edit mode will lead to your smoke appearing more than once during rendering, like a repeating texture.

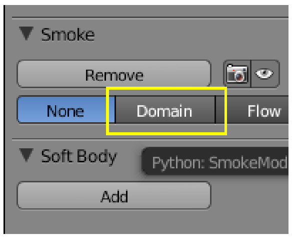
Make sure you're in object mode and go to the physics tab. Add smoke and chose the radio button labeled 'Domain'. For now that's all, we will return to the new settings that popped up later on.



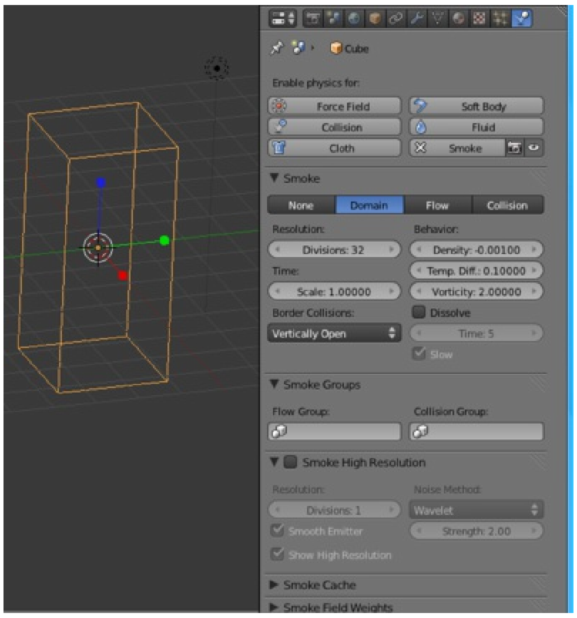
The physics tab might be hidden



Add smoke



Chose domain for the cube



Resolution: Divisions: 32
Behavior: Density: -0.00100
Temp. Diff.: 0.10000
Time: Scale: 1.00000
Vorticity: 2.00000
Border Collisions: Vertically Open
Dissolve
Time: 5
Slow
Flow Group:
Collision Group:
Resolution: Divisions: 1
Noise Method: Wavelet
Smooth Emitter
Strength: 2.00
Show High Resolution

The Smoke Domain Object

Generic options

Resolution
: How detailed the smoke is. A resolution of 32 will bake in a few seconds, while a resolution of 100 can take up to a half hour on most PC's.

Time Scale
: Affects how fast the simulation plays.

Border Collisions
: Vertically Open

  Smoke disappears when it collides with the top and bottom of the domain.

  Open

  Smoke disappears when it crosses the boundaries of the domain object.

  Collide All

  Domain Boundaries are treated as collision objects, the smoke will collide and stay inside.

Temperature and Density
: How much Density and Temperature affect smoke motion. Higher Values make faster-rising smoke.

Vorticity
: Affects how turbulence/rotation, or swirly the smoke is.

Dissolve
: Allow the smoke to dissipate over time.

Time
: The speed of the smoke's dissipation.

Slow
: Use 1/Time instead of Time, making the smoke dissolve slower.

Smoke Groups options

to do

Smoke High Resolution options

The High Resolution option lets you simulate at low resolution and then uses noise techniques to enhance the resolution without actually computing it. This allows animators to set up a low resolution simulation quickly and later add details without changing the overall fluid motion.

Various methods for this are available, including the default: Wavelet, which is an implementation of 'turb.php|Wavelet Turbulence for Fluid Simulation'

Resolution/Divisions
: Enhance the resolution of smoke by this factor using noise.

Smooth Emitter
: Smoothens emitted smoke to avoid blockiness.

Show High Resolution
: Show high resolution using amplification.

Noise Method
: Wavelet

  FFT

Strength
: Strength of noise.

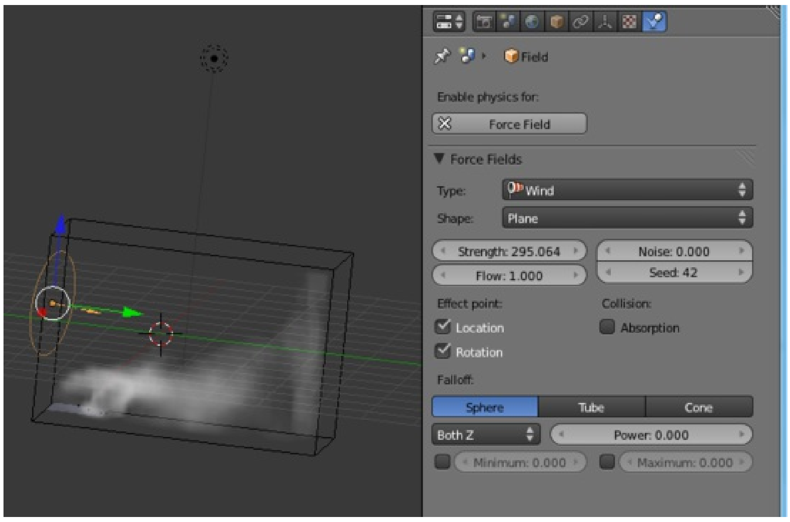
Smoke Field Weights options

Determines how much various forces and force fields affect the smoke.

Gravity
: How much the smoke is affected by Gravity.

All
: Changes the overall influence of all force fields.

The other settings determine how much various Force Fields affect the smoke.


Field

Enable physics for:

Force Field

Force Fields

Type: Wind

Shape: Plane

Strength: 295.064
Noise: 0.000

Flow: 1.000
Seed: 42

Effect point:
Location ✓
Rotation ✓

Collision:
Absorption

Falloff:
Sphere | Tube | Cone

Both Z
Power: 0.000

Minimum: 0.000
Maximum: 0.000

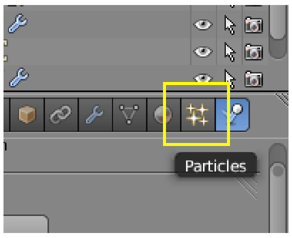
Smoke Flow object

Create a Flow Object

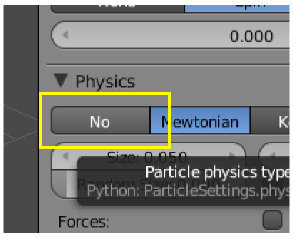
Once you have defined the volume that will contain smoke, we'll add an object from which the smoke will be emitted. Add another cube and make sure it's inside the domain cube ⇧ ShiftA » Mesh » Cube; 3D view must be selected).

While in edit mode go to physics and add smoke to the small cube, too. This time chose Flow.

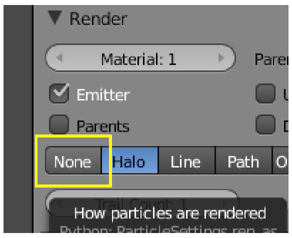
The smoke will not be emitted from the object itself but from particles the object emits. So we need to set up a particle system. With the small cube still selected go to the particle tab. Add a new particle system and turn off the physics because we want our smoke emit from stationary place. We also don't want to see the particles so turn off the render, too.



The particles tab is right next to the physics tab

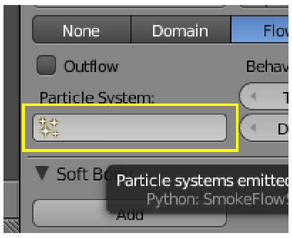


We don't want the particles to be affected by physics

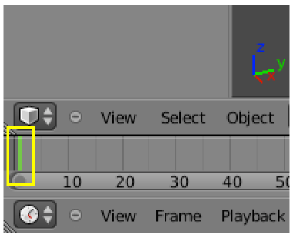


We also don't want to see the particles

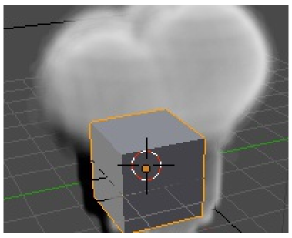
Now go back to the physics tab and chose the particle system in the smoke section. There should be a list with just one system to chose from that is called 'ParticleSystem' since we did not change the name. Now you can scrub through the timeline to see smoke coming from the cube. Another way to preview the smoke is starting the animation by AltA (stop it the same way).



Select the newly created particle system here



Either scrub on the timeline or use ALT+A


Now there should be smoke in the viewport

Settings

Outflow
    Delete smoke from simulation.
Particle System
    Particle system emitted from the object.
Initial Velocity
    Smoke inherits its velocity from the emitter particle.
Multiplier
    Multiplier to adjust velocity passed to smoke.

Initial Values

Absolute Density
    Only allow given density value in emitter area.
Density
    Initial density value.
Temp. Diff.
    Temperature to ambient temperar.

Collisions

Smoke can collide with mesh objects, using the 'Collision' option in smoke. Currently only static collision objects are supported.

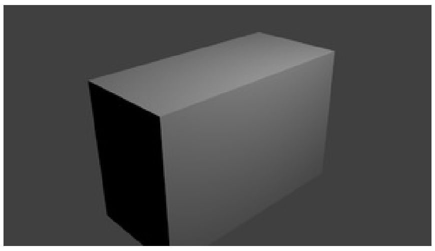
Forces

Blender's force fields (such as wind or vortex fields) are also supported, modifying the smoke simulation as they do for other physics systems such as particles.

Smoke Material

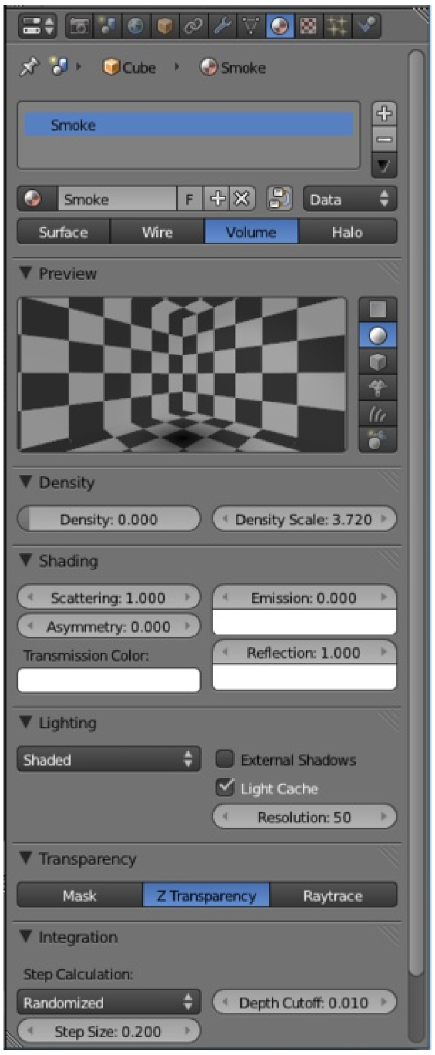
Create the Material

Simulating the smoke is easy, however rendering it is not.


The Render without the correct smoke material.

Rendering at this point will result in just the big cube (Image, F12) or in a crash (Animation, CtrlF12).

The material must be a volumetric material with a Density of 0, and a high Density Scale.


Cube > Smoke

Smoke

Smoke F Data

Surface | Wire | Volume | Halo

Preview

Density

Density: 0.000
Density Scale: 3.720

Shading

Scattering: 1.000
Emission: 0.000
Asymmetry: 0.000
Transmission Color:
Reflection: 1.000

Lighting

Shaded
External Shadows
Light Cache ✓
Resolution: 50

Transparency

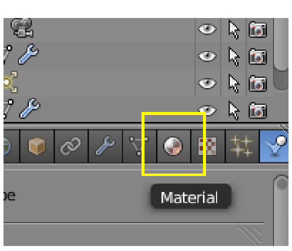
Mask | Z Transparency | Raytrace

Integration

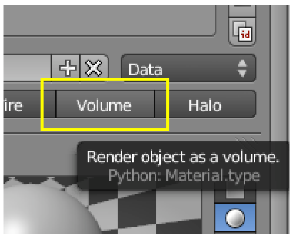
Step Calculation:
Randomized
Depth Cutoff: 0.010
Step Size: 0.200

The material settings.

The first issue can easily be fixed by working on the material and texture of the domain cube. Smoke requires a complex material to render correctly. Select the big cube and go to the material tab. There change the material to 'Volume' and set the density to 0. If you set the density to values bigger than 0 the domain cube will be filled with the volume material. The other settings will affect the smoke, though. We'll cover those later.

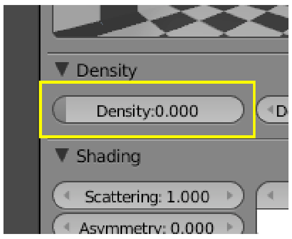


Go to the material tab



Render object as a volume.
Python: Material.type

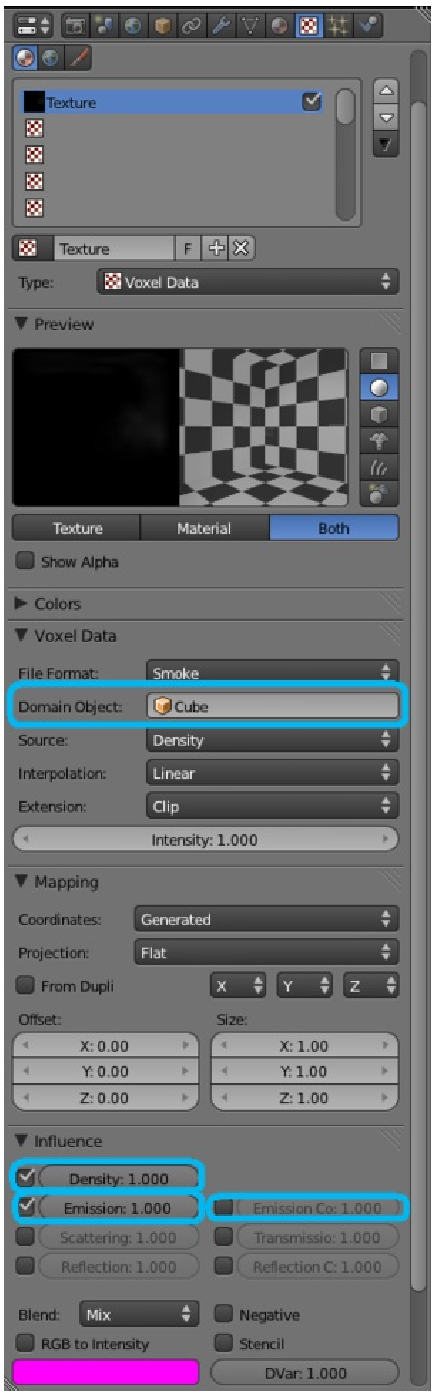
Smoke needs a volume material


Density
Density:0.000
Shading
Scattering: 1.000
Asymmetry: 0.000

Density applies to the cube only, so we need to set it to 0

Add the Texture

In addition, Smoke requires its own texture. Blender 2.5 has a new texture just for rendering smoke called Voxel Data. You must remember to set the domain object and change the influence.


Texture

Texture F

Type: Voxel Data

Preview

Texture | Material | Both

Show Alpha

Colors

Voxel Data

File Format: Smoke

Domain Object: Cube

Source: Density

Interpolation: Linear

Extension: Clip

Intensity: 1.000

Mapping

Coordinates: Generated

Projection: Flat

From Dupli X Y Z

Offset: X: 0.00 Y: 0.00 Z: 0.00

Size: X: 1.00 Y: 1.00 Z: 1.00

Influence

Density: 1.000

Emission: 1.000 Emission Co: 1.000

Scattering: 1.000 Transmissio: 1.000

Reflection: 1.000 Reflection C: 1.000

Blend: Mix Negative

RGB to Intensity Stencil

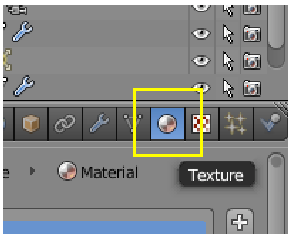
DVar: 1.000

The texture settings.

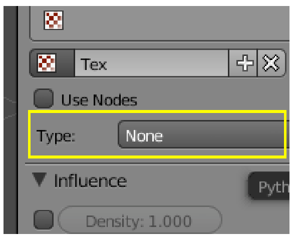
Go to the texture tab and change the type to 'Voxel Data'. Under the Voxel Data-Settings set the domain object to our domain cube (it should be listed just as 'Cube' since we are using Blender's default cube. Under Influence check 'Density' and leave it at 1.000 (Emission should be automatically checked, too). Now you should be able to render single frames. You can choose to color your smoke as well, by turning "Emmision Color" back on.

To see the smoke more clearly

Under the world tab, chose a very dark color for the horizon.



We need to add a texture of


Tex

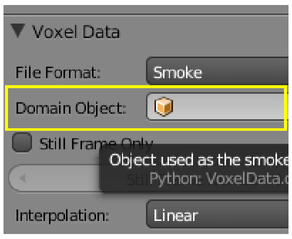
Use Nodes

Type: None

Influence

Density: 1.000

Type should be Voxel Data


Voxel Data

File Format: Smoke

Domain Object:

Still Frame Only

Object used as the smoke

Python: VoxelData.

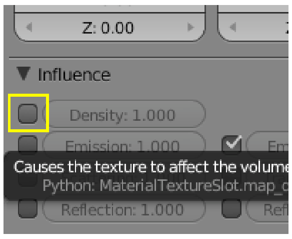
Interpolation: Linear

The domain is once again

the smoke

Type should be Voxel Data

our big cube

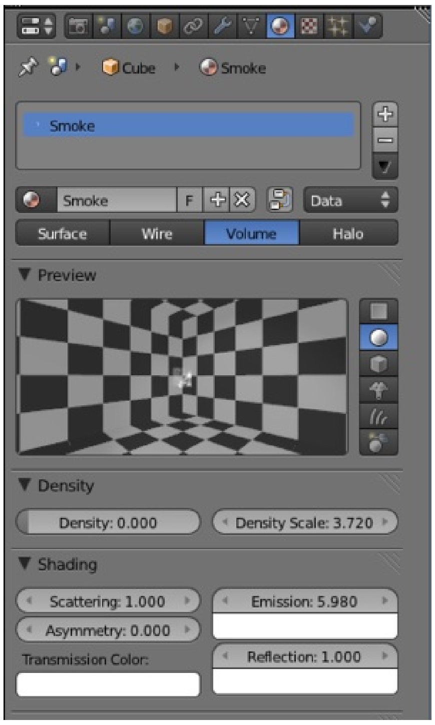

Z: 0.00
Influence
Density: 1.000
Emission: 1.000
Causes the texture to affect the volume
Python: MaterialTextureSlot.map_
Reflection: 1.000
Use density as influence

Finally your first smoke render :)

The rendered smoke. It's hard to see, but it's there.

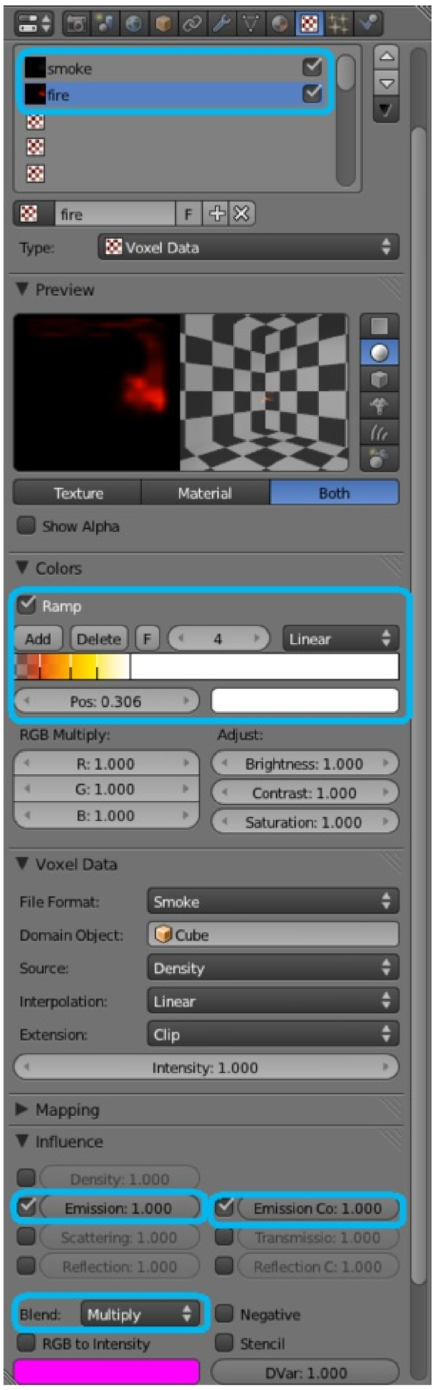
Extending the Smoke Simulator: Fire!

You can also turn your smoke into fire with another texture! To make fire, turn up the Emmision Value in the Materials panel.



The Fire material.

Then, add another texture (Keep the old texture or the smoke won't show). Give it a fiery color ramp- which colors based on the alpha, and change the influence to emmision and emmision color. Change the blend to Multiply.


smoke ✓
fire ✓

fire F
Type: Voxel Data

Preview
Texture | Material | [✓ Both]
Show Alpha

Colors
✓ Ramp
Add | Delete | F | 4 | Linear
Pos: 0.306

RGB Multiply:
R: 1.000
G: 1.000
B: 1.000

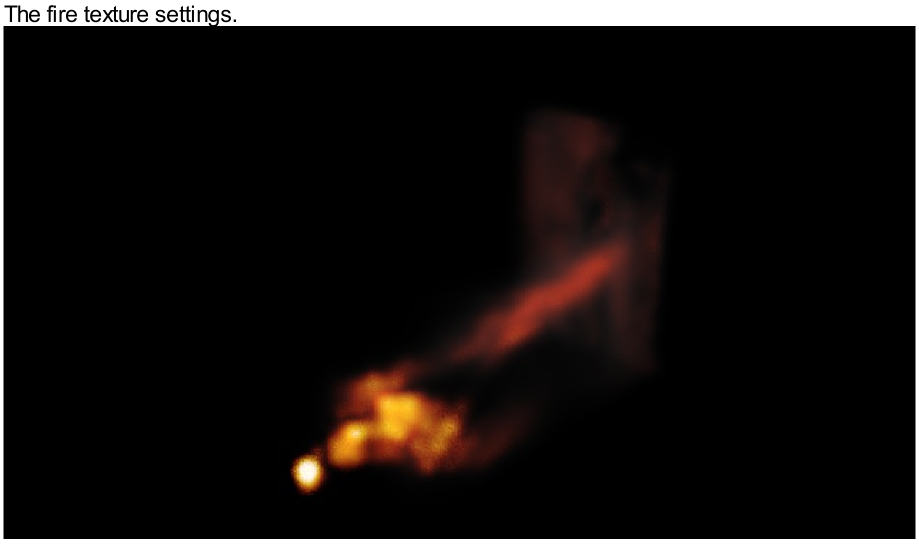
Adjust:
Brightness: 1.000
Contrast: 1.000
Saturation: 1.000

Voxel Data
File Format: Smoke
Domain Object: Cube
Source: Density
Interpolation: Linear
Extension: Clip
Intensity: 1.000

Mapping
Influence
Density: 1.000
✓ Emission: 1.000
✓ Emission Co: 1.000
Scattering: 1.000
Transmissio: 1.000
Reflection: 1.000
Reflection C: 1.000

Blend: Multiply
Negative
RGB to Intensity
Stencil
DVar: 1.000

The fire texture settings.

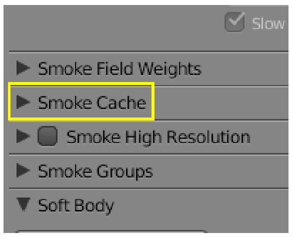

The fire render.

Baking Smoke Simulations

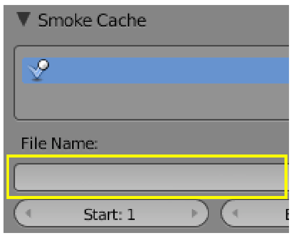
If you want to render an animation, you need to bake your smoke first. Baking is simply calculating a simulation. To bake the smoke, the file must be saved. The calculations are stored in a cache file which can be named. On occasion, Baking may crash Blender. [See troubleshooting]

By scrubbing through the timeline or running the animation in the viewport via AltA you already did some realtime-baking to the memory. But for rendering the animation, the baked data must be on disk. And before you can bake, you need to save your Blendfile.

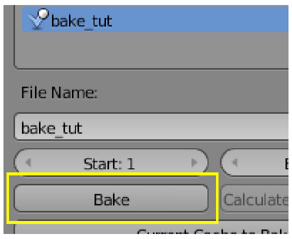
Next select the domain cube and go to the physics tab where you open the Smoke Cache section. Give your cache a file name by entering it into the text box and hitting ↵ Enter. By hitting Bake your simulation data is computed and stored to disk. Notice that the scrubbing-bug in the timeline is gone now? At this point you should be able to render the animation.



Go to the cache section of your smoke domain



The files on disk need a name



Finally we're ready to bake

Smoke Appendix

Troubleshooting

- **Q.** The smoke cache is greyed out!
- **A.** Save your .blend file.

- **Q.** Blender Crashes when I push the Domain Button of a smoke simulation!
- **A.** This is caused by outdated drivers. Update your Drivers.

- **Q.** Blender Crashes when Smoke Simulations bake!
- **A.** You ran out of RAM to compute it. Try Baking at a lower resolution.

- **Q.** The smoke isn't rendering!
- **A.** Go back and read the documentation.

- **Q.** When I try to make fire, it gives me strange results, or doesn't show up.
- **A.** Make sure you have a high emmision Value for the Material, and that you have a Smoke Density texture, and that you set the fire texture to Multiply.

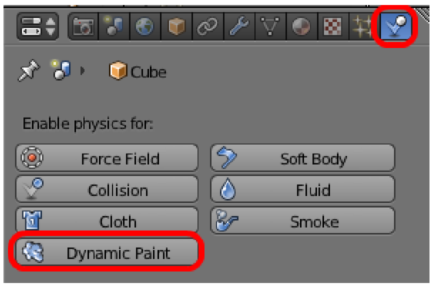
External links

- In-Depth introduction to smoke and fire in Blender 2.5 covering most of the pitfalls
- Guide to realistic fire in Blender by MiikaH

Dynamic Paint

Dynamic paint is a new modifier and physics system that can turn objects into paint canvases and brushes, creating vertex colors, image sequences or displacement. This makes many effects possible that were previously difficult to achieve, for example footsteps in the snow, raindrops that make the ground wet, paint that sticks to walls, or objects that gradually freeze.

This guide explains the very basics of Dynamic Paint user interface and general features.



How to activate the Dynamic Paint

Activating the modifier

Dynamic Paint can be activated from the "Physics" tab of the "Properties" editor.

Types

Modifier itself has two different types:

Canvas
: Makes object receive paint from Dynamic Paint brushes.

Brush
: Makes object apply paint on the canvas.

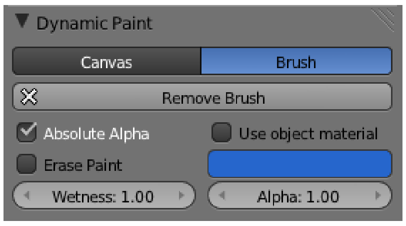
Note
You can also enable brush and canvas simultaneously. In that case same object's "brush" doesn't influence it's "canvas", but can still interact with other objects in the scene.

See also

- A step-by step introduction
- A detailed guide that covers every setting with images and examples (Currently not up-to-date)

Dynamic Paint Brush

Main Panel


Dynamic Paint
Canvas | Brush
Remove Brush
☑ Absolute Alpha ☐ Use object material
☐ Erase Paint
Wetness: 1.00 Alpha: 1.00

Brush main panel

From the first brush panel you can define how brush affects canvas color surfaces.

Absolute Alpha
: This setting limits brush alpha influence. Without it, brush is "added" on surface over and over again each frame, increasing alpha and therefore influence of brush on canvas. In many cases however, it's preferred to not increase brush alpha if it already is on brushes level.

Erase Paint
: Makes brush dissolve exiting paint instead of adding it.

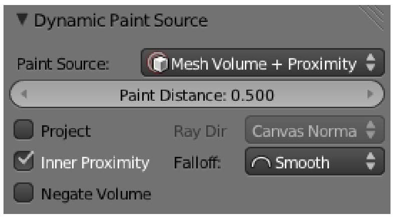
Wetness
: Defines how "wet" new paint is. Wetness is visible on "Paint" surface "wetmap". Speed of "Drip" and "Spread" effects also depends on how wet the paint is.

Use object material
: When enabled, you can define a material to be used as brush color. This includes material's base color and all textures linked to it, eventually matching the rendered diffuse color. This setting is only available when using "Blender Internal" renderer at the moment.

Otherwise you can define a color for the brush from the color box below.

Alpha
: Defines brush alpha or visibility. Final wetness is also affected by alpha.

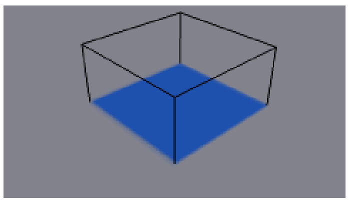
Source Panel


Dynamic Paint Source
Paint Source: Mesh Volume + Proximity
Paint Distance: 0.500
☐ Project Ray Dir Canvas Norma
☑ Inner Proximity Falloff: Smooth
☐ Negate Volume

Brush source panel

Brush "Source" setting lets you define how brush influence/intersection is defined.

There are currently five brush behavior types to choose from, each having individual settings for further tweaking:


Brush Source - Volume

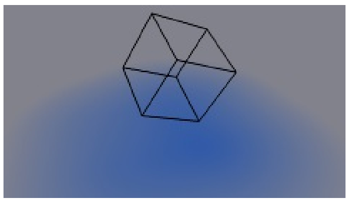
Mesh Volume

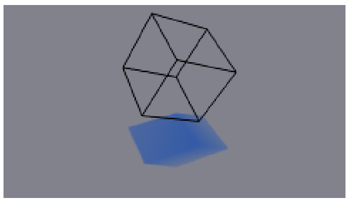
This the default option. Brush affects all surface point inside the mesh volume.

Proximity

Only uses defined distance to the closest point on brush mesh surface. Note that inside of the volume is not necessarily affected because it's not close to the surface.

Proximity falloff type can be "Smooth", "Sharp" or tweaked with a color ramp.


Brush Source - Proximity. Brush affects all canvas pixels around it


"Project" setting enabled. See how brush only affects canvas in normal

affects all canvas pixels around it　direction

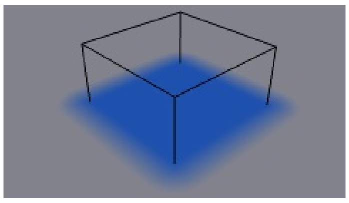
Project
: Projects brush to the canvas from a defined direction. Basically this can be considered as "direction aligned" proximity.

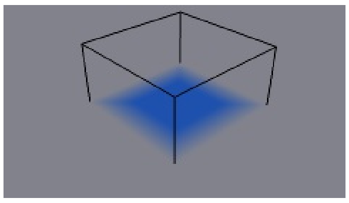
Mesh Volume + Proximity

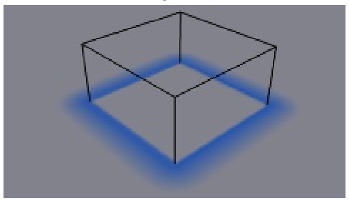
Same as volume type, but also has influence over defined distance. Same falloff types as for "Proximity" type are available.

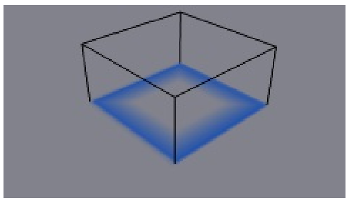
Inner Proximity
: Applies proximity inside the mesh volume.

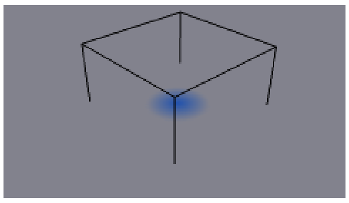
Negate Volume
: Negates brush alpha within mesh volume.


"Volume + Proximity" brush with no additional settings


Inner Proximity. Proximity falloff is now visible inside the volume

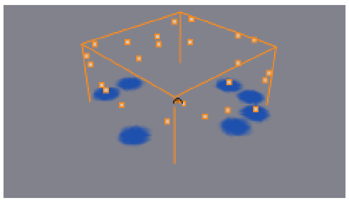

Negate Volume. Inner side of the volume has become completely transparent


Inner Proximity and Negate Volume enabled together


Brush Source - Object Center

Object Center

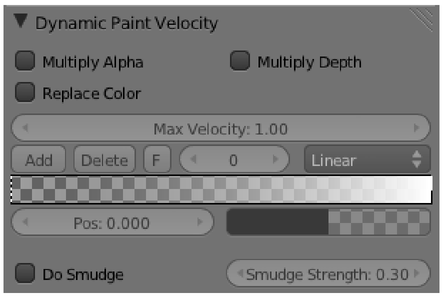
Instead of calculating proximity to the brush object mesh, which can be quite slow in some cases, only distance to only center is calculated. This is much faster and often good enough.


Brush Source - Particle System

Particle System

Brush influence is defined by particles from a selected particle system.

Velocity Panel


Dynamic Paint Velocity
Multiply Alpha　Multiply Depth
Replace Color
Max Velocity: 1.00
Add　Delete　F　0　Linear
Pos: 0.000
Do Smudge　Smudge Strength: 0.30

Velocity panel

This panel shows brush options that are based on object velocity.

On top you have a color ramp and several related settings. Basically the color ramp represents brush velocity values: left side being zero velocity and right side being the "Max velocity". Speed is measured in "Blender units per frame".

Tick boxes above can be used to define color ramp influence.

Multiply Alpha
: Uses color ramp's alpha value depending on current velocity and multiplies brush alpha with it.

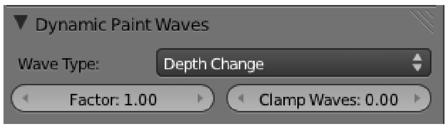
Replace Color
: Replaces the brush color with the ramp color.

Multiply Depth
: Multiplies brushes "depth intersection" effect. Basically you can adjust displace and wave strength depending on brush speed.

Smudge settings
: Enabling Smudge makes the brush "smudge" (or "smear") existing colors on the surface as it moves. The strength of this effect can be defined from the "Smudge Strength" property.

: Even when smudge is enabled brush still does it's normal paint effect. If you want a purely smudging brush use zero alpha. It's also possible to have "Erase" option enabled together with smudge.

Waves Panel



Brush Waves panel

This panel is used to adjust brush influence to "Wave" surfaces.

You can use "Wave Type" menu to select what effect this brush has on the wave simulation. Below are two settings for further adjustments.

Factor
: Adjusts how strongly brush "depth" affects the simulation. You can also use negative values to make brush pull water up instead of down.

Clamp Waves
: In some cases the brush goes very deep inside the surface messing whole simulation up. You can use this setting to "limit" influence to only certain depth.

There are four "Wave Type" options available:

Depth Change
: This option makes brush create waves when the intersection depth with the surface is *changed* on that point. If the brush remains still it won't have influence.

: Using a negative "Factor" with this type can create a nice looking "wake" for moving objects like ships.

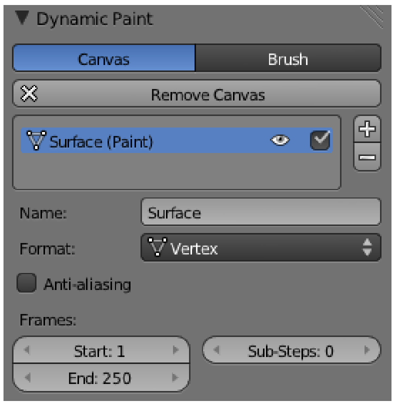
Obstacle
: Constantly affects surface whenever intersecting. Waves are also reflected off this brush type. However, due the nature of wave simulation algorithm this type creates an unnatural "dent" in the surface if brush remains still.

Force
: Directly affects the velocity of wave motion. Therefore the effect isn't one to one with brush intersection depth, yet the force strength depends on it.

Reflect Only
: This type has no visible effect on the surface alone but reflects waves that are already on the surface.

Dynamic Paint Canvas

Main Panel



Canvas main panel

The first panel of canvas contains the list of Dynamic Paint surfaces. These surfaces are basically layers of paint, that work independently from each other. You can define individual settings for them and bake them separately.

If surface type/format allows previewing results in 3D-viewport, an eye icon is visible to toggle preview.

The checkbox toggles whether surface is active at all. If not selected, no calculations or previews are done.

You can also give each surface an unique name to easily identify them.

Below you can set surface type and adjust quality and timing settings.

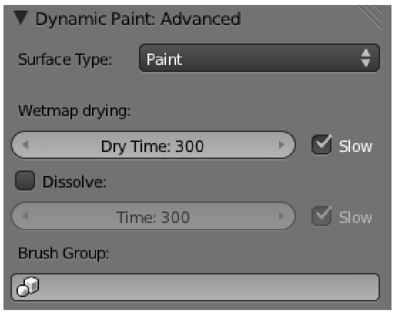
Each surface has a certain format and type. Format determines how data is stored and outputted. Currently there are two formats available:

- Image Sequences. Dynamic Paint generates UV wrapped image files of defined resolution as output.
- Vertex. Dynamic Paint operates directly on mesh vertex data. Results are stored by point cache and can be displayed in viewports. However, using vertex level also requires a highly subdivided mesh to work.

From quality settings you can adjust image resolution (for image sequences) and anti-aliasing.

Then you can define surface processing start and end frame, and number of used sub-steps. Sub-steps are extra samples between frames, usually required when there is a very fast brush.

Advanced Panel



Canvas advanced panel

From "Advanced" panel you can adjust surface type and related settings.

Each surface has a "type" that defines what surface is used for. Available types are:

- Paint
- Displace
- Waves
- Weight

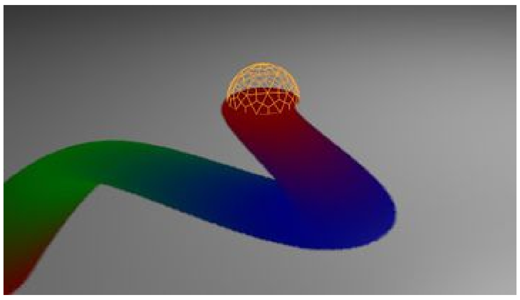
Common options

For each surface type there are special settings to adjust. Most types have the settings *Dissolve* and *Brush*:

Dissolve
: used to make the surface smoothly return to its original state during a defined time period

Brush Group
: used to define a specific object group to pick brush objects from

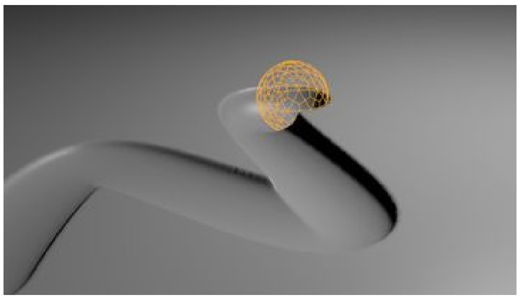
Paint


Paint Surface

"Paint" is the basic surface type that outputs color and wetness values. In case of vertex surfaces results are outputted as vertex colors.

Wetmap is a black-and-white output that visualizes paint wetness. White being maximum wetness, black being completely dry. It is usually used as mask for rendering. Some "paint effects" affect wet paint only.

Displace

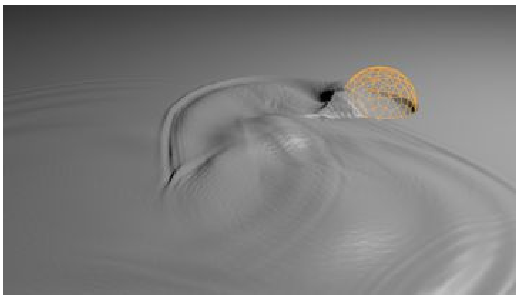

Displace Surface

This type of surface outputs intersection depth from brush objects.

Tip

If the displace output seems too rough it usually helps to add a "Smooth" modifier after Dynamic Paint in the modifier stack.

Waves


Waves Surface

This surface type produces simulated wave motion. Like displace, wave surface also uses brush intersection depth to define brush strength.

You can use following settings to adjust the motion:

Open Borders
: Allows waves to pass through mesh "edges" instead of reflecting from them.

Timescale
: Directly adjusts simulation speed without affecting simulation outcome. Lower values make simulation go slower and otherwise.

Speed
: Affects how fast waves travel on the surface. This setting is also corresponds to the size of the simulation. Half the speed equals surface double as large.

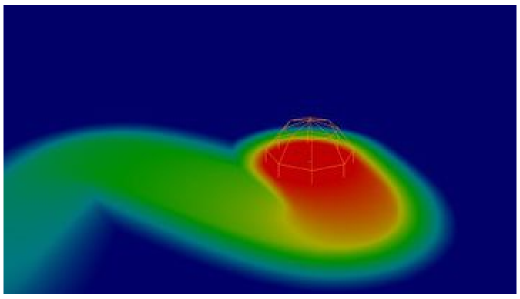
Damping
: Reduces the wave strength over time. Basically adjusts how fast wave disappears.

Spring
: Adjusts the force that pulls water back to "zero level".

Tip

In some cases the wave motion gets very unstable around brush. It usually helps to reduce wave speed, brush "wave factor" or even the resolution of mesh/surface.

Weight

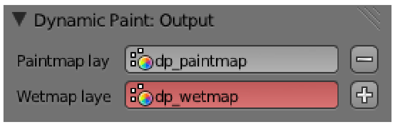

Weight Surface

This is a special surface type only available for vertex format. It outputs vertex weight groups that can be used by other Blender modifiers and tools.

Tip

It's usually preferred to use "proximity" based brushes for weight surfaces to allow smooth falloff between weight values.

Output Panel



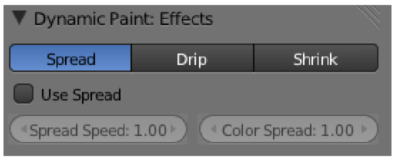
Canvas output panel

From "Output" panel you can adjust how surface outputs its results.

For "Vertex" format surfaces, you can select a mesh data layer (color / weight depending on surface type) to generate results to. You can use the "+"/"-" icons to add/remove a data layers of given name. If layer with given name isn't found, it's shown as red.

For "Image Sequence" surfaces, you can define used "UV Layer" and output file saving directory, filenames and image format.

Effects Panel



Canvas effects panel

This is a special feature for "Paint" type surface. It generates animated movement on canvas surface.

Currently there are 3 effects available:

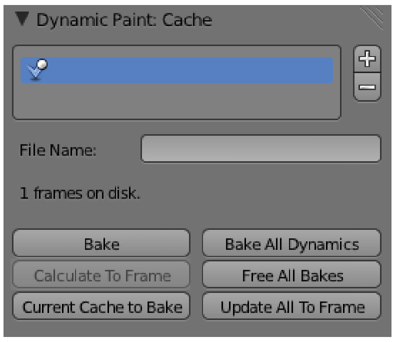
Spread
: Paint slowly spreads to surrounding points eventually filling all connected areas.

Drip
: Paint moves in specific direction specified by Blender force fields, gravity and velocity with user defined influences.

Shrink
: Painted area slowly shrinks until disappears completely.

For spread and drip effects, only "wet paint" is affected, so as the paint dries, movement becomes slower until it stops.

Cache Panel



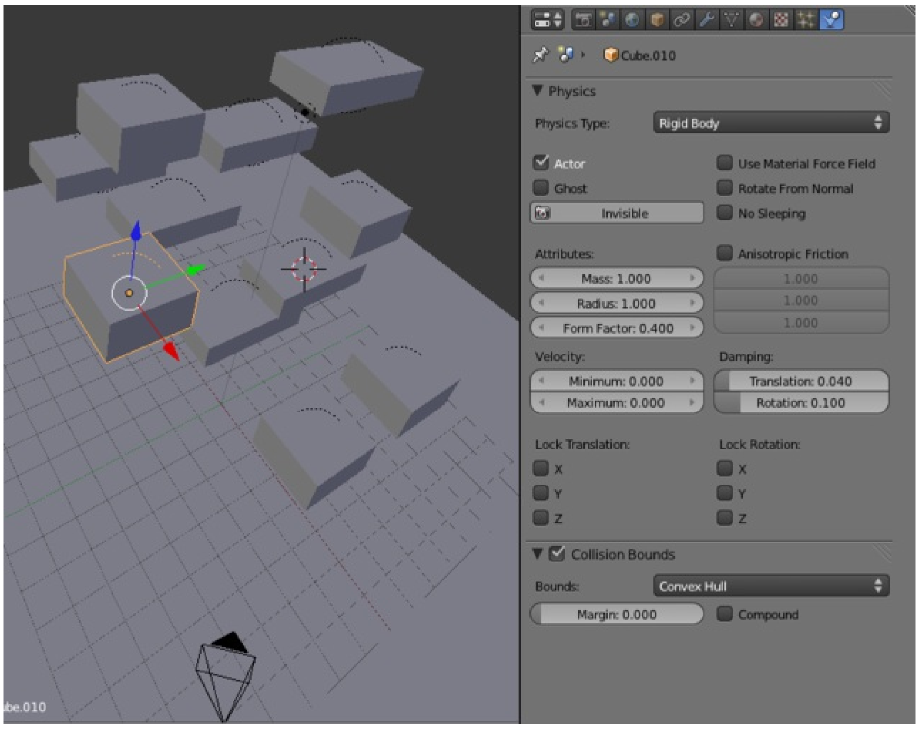
Canvas cache panel

This panel is currently only visible for "vertex" format surfaces. You can use it to adjust and bake point cache.

Converting Game Engine Physics

Sometimes, you may want to animate a wall being broken down by an object, or a bunch of objects collapsing, falling, or bouncing with accurate physics. You could manually insert keyframes and do trial and error adjusting with F-Curves to simulate physics and acceleration, or, you can do it much easier and automatically by taking advantage of Blender Game Engine Physics. Blender now has a feature which allows you to record animation in a blender game and turn it into Blender Animation Keyframes.

Animation can be recorded by going Game>Record Animation. The animation can them to recorded with Alt+A



Cube.010

Physics

Physics Type: Rigid Body

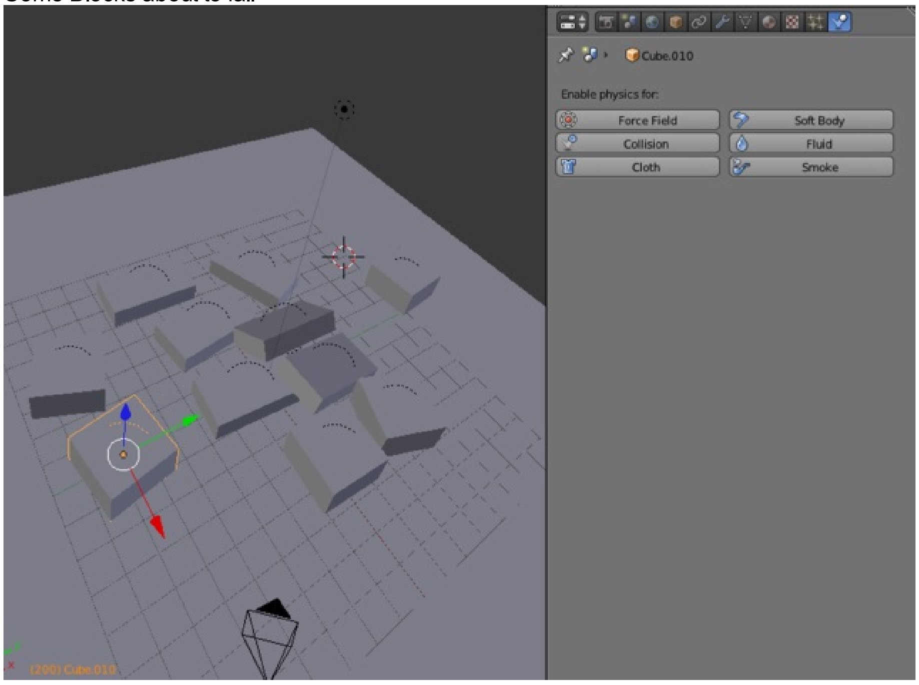
✓ Actor	Use Material Force Field
Ghost	Rotate From Normal
Invisible	No Sleeping
Attributes:	Anisotropic Friction
Mass: 1.000	1.000
Radius: 1.000	1.000
Form Factor: 0.400	1.000
Velocity:	Damping:
Minimum: 0.000	Translation: 0.040
Maximum: 0.000	Rotation: 0.100
Lock Translation:	Lock Rotation:
X	X
Y	Y
Z	Z

✓ Collision Bounds

Bounds: Convex Hull

Margin: 0.000 Compound

Some Blocks about to fall



Cube.010

Enable physics for:

Force Field	Soft Body
Collision	Fluid
Cloth	Smoke

A Pile

If you just want a static pile of stuff, you can move to the last frame, and delete all the keyframes quickly by turning them into NLAs and deleting.

Simulation performance adjustments

OpenMP (Mac OSX)

How to use the distributed applescript to optimize simulation performance on OSX

Suboptimal baking performance

Often you will encounter suboptimal baking performance with openMP (OMP) multithreaded simulations, especially fluids.

This is due how the domain splitting distributes threads to cells which are sometimes not "filled" whereas calculations, resulting in lot of threads not giving any speedboost. This is almost dependent on the resolution the simulation and object density.

If you have such an "undersaturated" simulation, it helps a lot to just reduce the OMP threads to a low number instead of letting OMP just use all available cores.

Solution

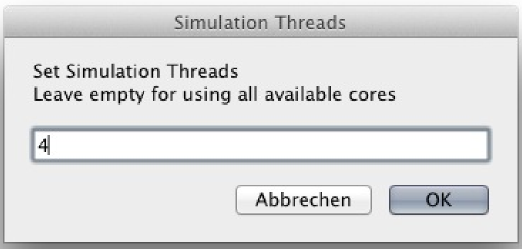
For OSX openMP-enabled Blender you can now use a delivered applescript to tune the OMP-threads used. This makes usage of the terminal obsolete.

The default is for now set to 4 cores.

If you leave the input field empty, the script gets the corecount of your logical cpu-cores (including HyperTrading) This is the same what openMP would pull without the environment variable set.

Steps to use the applescript

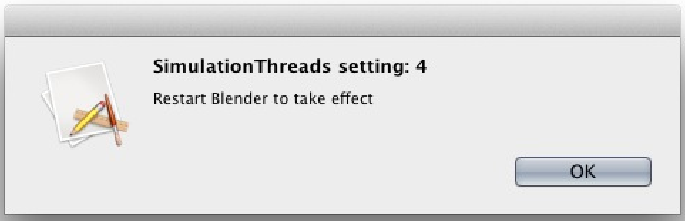
1. In your Blenderfolder open the "set_simulation_threads" applescript set_simulation_threads
2. Set the threadcount you want to use (leave empty to let OMP get all available cores)


Simulation Threads

Set Simulation Threads
Leave empty for using all available cores

4

3. Press o.k. to set the new value, a messagebox will show you the new setting accepted.


SimulationThreads setting: 4
Restart Blender to take effect

4. Close and reopen Blender to take over the setting.
5. Watch your baking progress or simulation framerates and perhaps repeat steps 1- 4 to get the optimal value.

Motion Tracking

Introduction

Motion tracking is a new technique available in Blender. It is still under development, and as of August 2013 supported basic operations for 2D motion tracking, 3D motion tracking, and camera solution. It's already ready to be used in production, as validated by "Tears of Steel."

Getting started

Motion tracking is included with the Blender 2.61 release and later versions. It's enabled by default for all platforms and can be used "out-of-the-box".

Here's brief descriptions of motion tracking tools currently available in Blender

Supervised 2D tracking

There's no common algorithm which can be used for all kinds of footage, feature points and their motions. Such algorithms can be constructed, but they'll be really slow and they can still fail, so the only way to perform 2D tracking is to manually choose the tracking algorithm and its settings. Current defaults work nicely for general footage which isn't very blurry and where feature points aren't getting highly deformed by perspective.

Improving 2D tracking is already in our TODO list, but it's not high priority at this moment. If you aren't sure about algorithms and settings and don't want to read this document, you can just play with settings and find one which works for you.

Manual lens calibration using grease pencil and/or grid

All cameras record distorted video. Nothing can be done about this because of the manner in which optical lenses work. For accurate camera motion, the exact value of the focal length and the "strength" of distortion are needed.

Currently, focal length can be automatically obtained only from the camera's settings or from the EXIF information -- there are no tools inside Blender which can estimate it. But there are some tools which can help to find approximate values to compensate for distortion. There are also fully manual tools where you can use a grid which is getting affected by distortion model and deformed cells defines straight lines in the footage. You can also use the grease pencil for this - just draw line which should be straight on the footage using poly line brush and adjust distortion values to make the grease pencil match lines on the footage.

To calibrate your camera more accurately, use the grid calibration tool from OpenCV. OpenCV is using the same distortion model, so it shouldn't be a problem.

Camera motion solving

Despite the fact that there's no difference in solving camera motion and object motion from a mathematical point of view, only camera solving is currently supported. And it still has some limitations, like unsupported solve of tripod motions or dominant plane motions (where all trackable features belong to one plane). These limitations are planned to be solved in the future.

Basic tools for scene orientation and stabilization

After solve, you need to orient the real scene in the 3D scene for more convenient compositing. There are tools to define the floor, the scene origin, and the X/Y axes to perform scene orientation.

If something is needed to stabilize video from the camera to make the final result looks nicer, 2D stabilization is available to help. It stabilizes video from the camera, which can compensate for camera jumps and tilt.

Basic nodes for compositing scene into real footage

Some new nodes were added to the Compositor to composite scene into footage in easier way. So there are nodes for 2D stabilization, distortion and undistortion which are easy to use.

Not implemented tools

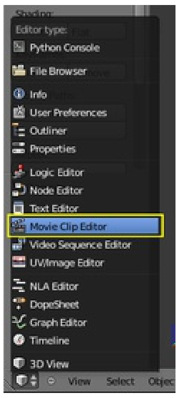
Some tools aren't available in Blender yet, but they are in our TODO list. So there's currently no support for such things as rolling shutter filtering, object motion solving, motion capturing. But you can try to hack this stuff using currently implemented things.

Manual

Movie Clip Editor

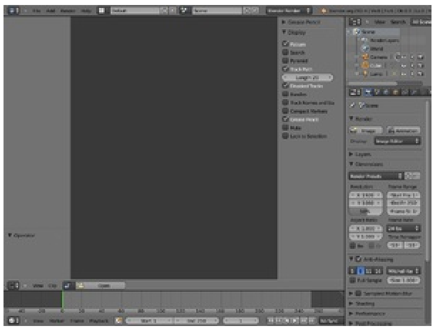
Almost all motion tracking tools are concentrated in the Movie Clip Editor. Currently it doesn't have any tools which aren't related to motion tracking, but in the future it may be expanded to be used for masking, so that's why it's given this more abstract name, rather than motion tracking.

This editor can be found in the list of editor types.



Editor type menu

When you switch to Movie Clip Editor window, the interface changes in the following way.

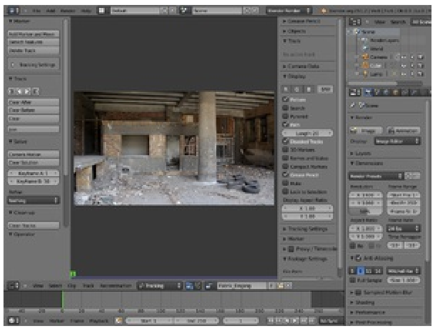

Movie Clip Editor interface

The next logical step to open a new video clip to start working with. There are several ways to to this:

- Use Open button from movie editor header
- Use Clip » Open menu
- Use AltO> shortcut

Both movie files and image sequences can be used in the clip editor. If you're using an image sequence there's one limitation on naming of files: the numbers at the end of the image name should be increasing continuously.

So, when a movie clip is loaded into the clip editor, extra panels are displayed in the interface.

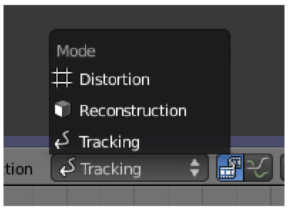

Movie Clip Editor with opened clip

There are plenty of new tools on the screen and here's short description of all of them.

First of all, it should be mentioned that the camera solver consists of three quite separate steps:

- 2D tracking of footage
- Camera intrinsics (focal length, distortion coefficients) specification/estimation/calibration
- Solving camera, scene orientation, scene reconstruction

Tools in the clip editor are split depending on which step they're used in, so the interface isn't cluttered up with scene orientation tools when only 2D tracking can be done. The currently displayed tool category can be changed using the Mode menu which is in the editor header.



Movie Clip Editor mode menu

But almost all operators can be called from menus, so it's not necessary to change the mode every time you want to use a tool which is associated with a different editor mode.

In tracking mode only tools which are related to tracking and camera solving are displayed. Camera solving tools are included here because it's after solving you'll most probably want to re-track existing tracks or place new tracks to make solving more accurate.

Tools available in tracking mode

Marker panel

- The **Add Marker and Move** operator places a new marker at the position of the mouse (which is under the button in this case, not ideal but it's just how things work) and then it can be moved to the needed location. When it's moved to the desired position, LMB can be used to finish placing the new marker. Also, ↵ Enter and Space can be used to finish placing the marker.

  But it's faster to use Ctrl LMB to place markers directly on the footage. This shortcut will place the marker in the place you've clicked. One more feature here: until you've released the mouse button, you can adjust the marker position by moving the mouse and using the track preview widget to control how accurately the marker is placed.

- The **Detect Features** operator detects all possible features on the current frame and places markers at these features. This operator doesn't take into account other frames, so it can place markers on features which belong to moving objects, and if camera is turning away from this shot, no markers would be placed on frames after the camera moved away.

  There are several properties for this operator:

  > **Placement** is used to control where to place markers. By default, they'll be added through the whole frame, but you can also outline some areas with interesting features with grease pencil and place markers only inside the outlined area. That's how the "Inside Grease Pencil" placement variant works. You can also outline areas of no interest (like trees, humans and so) and place markers outside of these areas. That's how the "Outside Grease Pencil" placement variant works.
  > **Margin** controls the distance from the image boundary for created markers. If markers are placed too close to the image boundary, they'll fail to track really quickly and they should be deleted manually. To reduce the amount of manual clean-up, this parameter can be used.
  > **Trackability** limits minimal trackability for placing markers. This value comes from the feature detection algorithm and basically it means: low values means most probably this feature would fail to track very soon, high value means it's not much such track. Amount of markers to be added can be controlled with this value.
  > **Distance** defines the minimal distance between placed markers. It's needed to prevent markers from being placed too close to each other (such placement can confuse the camera solver).

- **Delete Track** is a quite self-explaining operator which deletes all selected tracks.

Track panel

- The first row of buttons is used to perform tracking of selected tracks (i.e. following the selected feature from frame to frame). Tracking can happen (in order of buttons):
  - Backward one frame
  - Backward along the sequence
  - Forward along the whole sequence
  - Forward one frame

  This operator depends on settings from the Tracking Settings panel, which will be described later.
  If during sequence tracking the algorithm fails to track some markers, they'll be disabled and tracking will continue for the rest of the markers. If the algorithm fails when tracking frame-by-frame, the marker is not disabled, and the most likely position of the feature on the next frame is used.

- **Clear After** deletes all tracked and keyframed markers after the current frame for all selected tracks.
- **Clear Before** deletes all tracked and keyframed markers before the current frame for all selected tracks.
- **Clear** clears all markers except the current one from all selected tracks.
- **Join** operator joins all selected tracks into one. Selected tracks shouldn't have common tracked or keyframed markers at the same frame.

Solve panel

Camera Motion operator solves the motion of camera using all tracks placed on the footage and two keyframes specified on this panel. There are some requirements:

- There should be at least 8 common tracks on the both of the selected keyframes.
- There should be noticeable parallax effects between these two keyframes.

If everything goes smoothly during the solve, the average reprojection error is reported to the information space and to the clip editor header. Reprojeciton error means the average distance between reconstructed 3D position of tracks projected back to footage and original position of tracks. Basically, reprojection error below 0.3 means accurate reprojection, 0.3-3.0 means quite nice solving which still can be used. Values above 3 means some tracks should be tracked more accurately, or that values for focal length or distortion coefficients were set incorrectly.

The **Refine** option specifies which parameters should be refined during solve. Such refining is useful when you aren't sure about some camera intrinsics, and solver should try to find the best parameter for those intrinsics. But you still have to know approximate initial values - it'll fail to find correct values if they were set completely incorrectly initially.

Cleanup Panel

This panel contains a single operator and its settings. This operator cleans up bad tracks: tracks which aren't tracked long enough or which failed to reconstruct accurately. Threshold values can be specified from sliders below the button. Also, several actions can be performed for bad tracks:

- They can simply be selected
- Bad segments of tracked sequence can be removed
- The whole track can be deleted

Clip Panel

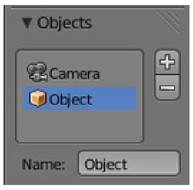
This panel currently contains the single operator Set as background which sets the clip currently being edited as the camera background for all visible 3D viewports. If there's no visible 3D viewports or the clip editor is open in full screen, nothing will happen.

Properties available in tracking mode

Grease Pencil Panel

It's a standard grease pencil panel where new grease pencil layers and frames can be controlled. There's one difference in the behavior of the grease pencil from other areas - when a new layer is created "on-demand" (when making a stroke without adding a layer before this) the default color for the layer is set to pink. This makes the stroke easy to notice on all kinds of movies.

Objects Panel



Objects Panel in clip editor

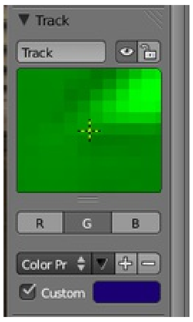
This panel contains a list of all objects which can be used for tracking, camera or object solving. By default there's only one object in this list which is used for camera solving. It can't be deleted and other objects can't be used for camera solving; all added objects are used for object tracking and solving only. These objects can be referenced from Follow Track and Object Solver constraints. Follow Track uses the camera object by default.

New objects can be added using + and the active object can be deleted with the - button. Text entry at the bottom of this panel is used to rename the active object.

If some tracks were added and tracked to the wrong object, they can be copied to another object using Track » Copy Tracks and Track » Paste Tracks.

The usage for all kind of objects (used for camera and object tracking) is the same: track features, set camera data, solve motion. Camera data is sharing between all objects and refining of camera intrinsics happens when solving camera motion only.

Track Panel



Track Panel in clip editor

First of all, track name can be changed in this panel. Track names are used for linking tracking data to other areas, like a Follow Track constraint.

The next thing which can be controlled here is the marker's enabled flag (using the button with the eye icon). If a marker is disabled, its position isn't used either by solver nor by constraints.

The button with the lock icon to the right of the button with the eye controls whether the track is locked. Locked tracks can't be edited at all. This helps to prevent accidental changes to tracks which are "finished" (tracked accurate along the whole footage).

The next widget in this panel is called "Track Preview" and it displays the content of the pattern area. This helps to check how accurately the feature is being tracked (controlling that there's no sliding off original position) and also helps to move the track back to the correct position. The track can be moved directly using this widget by mouse dragging.

If an anchor is used (the position in the image which is tracking is different from the position which is used for parenting), a preview widget will display the area around the anchor position. This configuration helps in masking some things when there's no good feature at position where the mask corner should be placed. Details of this technique will be written later.

There's small area below the preview widget which can be used to enlarge the vertical size of preview widget (the area is highlighted with two horizontal lines).

The next setting is channels control. Tracking happens in gray-scale space, so a high contrast between the feature and its background yields more accurate tracking. In such cases disabling some color channels can help.

The last thing is custom color, and the preset for it. This setting overrides the default marker color used in the clip editor and 3D viewport, and it helps to distinguish different type of features (for example, features in the background vs. foreground and so on). Color also can be used for "grouping" tracks so a whole group of tracks can be selected by color using the Select Grouped operator.

- **Marker Tip 1:** To select good points for tracking, use points in the middle of the footage timeline and track backwards and forwards from there. This will provide a greater chance of the marker and point staying in the camera shot.

Camera Data Panel

This panel contains all settings of the camera used for filming the movie which is currently being edited in the clip editor.

First of all, predefined settings can be used here. New presets can be added or unused presets can be deleted. But such settings as distortion coefficients and principal point aren't included into presets and should be filled in even if camera presets are used.

- **Focal Length** is self-explanatory; it's the focal length with which the movie was shot. It can be set in millimeters or pixels. In most cases focal length is given in millimeters, but sometimes (for example in some tutorials on the Internet) it's given in pixels. In such cases it's possible to set it directly in the known unit.
- **Sensor Width** is the width of the CCD sensor in the camera. This value can be found in camera specifications.
- **Pixel Aspect Ratio** is the pixel aspect of the CCD sensor. This value can be found in camera specifications, but can also be guessed. For example, you know that the footage should be 1920x1080, but the images themselves are 1280x1080. In this case, the pixel aspect is:

```
1920 / 1280 = 1.5
```

- **Optical Center** is the optical center of the lens used in the camera. In most cases it's equal to the image center, but it can be different in some special cases. Check camera/lens specifications in such cases. To set the optical center to the center of image, there's a Center button below the sliders.
- **Undistortion K1, K2 and K3** are coefficients used to compensate for lens distortion when the movie was shot. Currently these values can be tweaked by hand only (there are no calibration tools yet) using tools available in Distortion mode. Basically, just tweak K1 until solving is most accurate for the known focal length (but also take grid and grease pencil into account to prevent "impossible" distortion).

Display Panel

This panel contains all settings which control things displayed in the clip editor.

- **R, G, B** and **B/W** buttons at the top of this panel are used to control color channels used for frame preview and to make the whole frame gray scale. It's needed because the tracking algorithm works with gray-scale images and it's not always obvious to see which channels disabled will increase contrast of feature points and reduce noise.
- **Pattern** can be used to disable displaying of rectangles which correspond to pattern areas of tracks. In some cases it helps to make the clip view cleaner to check how good tracking is.
- **Search** can be used to disable displaying of rectangles which correspond to search areas of tracks. In some cases it helps to make the clip view cleaner to check how good tracking is. Only search areas for selected tracks will be displayed.
- **Pyramid** makes the highest pyramid level be visible. Pyramids are defined later in the Tracking Settings panel section, but basically it helps to determine how much a track is allowed to move from one frame to another.
- **Track Path** and **Length** control displaying of the paths of tracks. The ways tracks are moving can be visible looking at only one frame. It helps to determine if a track jumps from its position or not.
- **Disabled Tracks** makes it possible to hide all tracks which are disabled on the current frame. This helps to make view more clear, to see if tracking is happening accurately enough.
- **Bundles** makes sense after solving the movie clip, and it works in the following way: the solved position of each track gets projected back to the movie clip and displayed as a small point. The color of the point depends on the distance between the projected coordinate and the original coordinate: if they are close enough, the point is green, otherwise it'll be red. This helps to find tracks which weren't solved nicely and need to be tweaked.
- **Track Names and Status** displays information such as track name and status of the track (if it's keyframed, disabled, tracked or estimated). Names and status for selected tracks are displayed.
- **Compact Markers**. The way in which markers are displayed (black outline and yellow foreground color) makes tracks visible on all kind of footage (both dark and light). But sometimes it can be annoying and this option will make the marker display more compactly - the outline is replaced by dashed black lines drawn on top of the foreground, so that marker areas are only 1px thick.
- **Grease pencil** controls if grease pencil strokes are allowed to be displayed and made.
- **Mute** changes displaying on movie frame itself with black square, It helps to find tracks which are tracked inaccurately or which weren't tracked at all.
- **Grid** (available in distortion mode only) displays a grid which is originally orthographic, but os affected by the distortion model. This grid can be used for manual calibration - distorted lines of grids are equal to straight lines in the footage.
- **Manual Calibration** (available in distortion mode only) applies the distortion model for grease pencil strokes. This option also helps to perform manual calibration. A more detailed description of this process will be added later.

- **Stable** (available in reconstruction mode only). This option makes the displayed frame be affected by the 2D stabilization settings. It's only a preview option, which doesn't actually change the footage itself.
- **Lock to Selection** makes the editor display selected tracks at the same screen position along the whole footage during playback or tracking. This option helps to control the tracking process and stop it when the track is starting to slide off or when it jumped.
- **Display Aspect Ratio** changes the aspect ratio for displaying only. It does not affect the tracking or solving process.

Tracking Settings Panel

Common options

This panel contains all settings for the 2D tracking algorithms. Depending on which algorithm is used, different settings are displayed, but there are a few that are common for all tracker settings:

Adjust Frames controls which patterns get tracked; to be more precise, the pattern from which frame is getting tracked. Here's an example which should make things clearer.

The tracker algorithm receives two images inside the search area and the position of a point to be tracked in the first image. The tracker tries to find the position of that point from the first image in the second image.

Now, this is how tracking of the sequence happens. The second image is always from a frame at which the position of marker isn't known (next tracking frame). But a different first image (instead of the one that immediately precedes the second image in the footage) can be sent to the tracker. Most commonly used combinations:

- An image created from a frame on which the track was keyframed. This configuration prevents sliding from the original position (because the position which best corresponds to the original pattern is returned by the tracker), but it can lead to small jumps and can lead to failures when the feature point is deformed due to camera motion (perspective transformation, for example). Such a configuration is used if **Adjust Frames** is set to 0.

- An image created from the current frame is sent as first image to the tracker. In this configuration the pattern is tracking between two neighboring frames. It allows dealing with cases of large transformations of the feature point but can lead to sliding from the original position, so it should be controlled. Such a configuration is used if **Adjust Frames** is set to 1.

If **Adjust Frames** is greater than 1, the behavior of tracker is: keyframes for tracks are creating every **Adjust Frames** frames, and tracking between keyframed image and next image is used.

Speed can be used to control the speed of sequence tracking. This option doesn't affect the quality of tracking; it just helps to control if tracking happens accurately. In most cases tracking happens much faster than real time, and it's difficult to notice when a track began to slide out of position. In such cases **Speed** can be set to Double or Half to add some delay between tracking two frames, so slide-off would be noticed earlier and the tracking process can be cancelled to adjust positions of tracks.

Frames Limit controls how many frames can be tracked when the Track Sequence operator is called. So, each Track Sequence operation would track maximum **Frames Limit** frames. This also helps to notice slide-off of tracks and correct them.

Margin can be used disable tracks when they become too close to the image boundary. This slider sets "too close" in pixels.

KLT tracker options

The KLT tracker is the algorithm used by default. It allows tracking most kinds of feature points and their motion. It uses pyramid tracking which works in the following way. The algorithm tracks an image larger than the defined pattern first to find the general direction of motion. Then it tracks a slightly smaller image to refine the position from the first step and make the final position more accurate. This iterates several times. The number of steps of such tracking is equal to the **Pyramid Level** option and we tell that on first step tracking happens for highest pyramid level. So Pyramid Level=1 is equal to pattern itself, and each next level doubles tracking image by 2.

The search area should be larger than the highest pyramid level and the "free space" between the search area and highest pyramid level defines how much the feature can move from one frame to another and still be tracked.

Default settings should work in most general cases, but sometimes the pyramid level should be changed. For example, when footage is blurry, adding extra pyramid levels helps to track them.

This algorithm can fail in situations where a feature point is moving in one direction and the texture around that feature point is moving in another direction.

SAD tracker options

On each step, the SAD tracker reviews the whole search area and finds the pattern on the second image which is most like the pattern which is getting tracking. This works pretty quickly, but can fail in several cases. For example, when there's another feature point which looks like the tracking feature point in the search area. In this case, SAD will tend to jump off track from one feature to another.

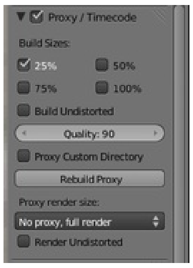
Correlation defines the threshold value for correlation between two patterns which is still considered successful tracking. 0 means there's no correlation at all, 1 means correlation is full.

There's one limitation: currently: it works for features of size 16x16 pixels only.

Marker Panel

This panel contains numerical settings for marker position, pattern and search area dimensions, and offset of anchor point from pattern center. All sliders are self-explanatory.

Proxy / Timecode Panel



☑ 25% ☐ 50%
☐ 75% ☐ 100%
☐ Build Undistorted
Quality: 90
☐ Proxy Custom Directory
No proxy, full render
☐ Render Undistorted

Proxy / Timecode Panel in clip editor

This panel contains options used for image proxies and timecodes for movies.

Proxy allows displaying images with lower resolution in the clip editor. This can be helpful in cases when tracking of 4K footage is happening on a machine with a small amount of RAM.

The first four options are used to define which resolutions of proxy images should be built. Currently it's possible to build images 25%, 50%, 75% and 100% of the original image size. Proxy size of 100% can be used for movies which contain broken frames which can't be decoded.

Build Undistorted means that the proxy builder also creates images from undistorted original images for the sizes set above. This helps provide faster playback of undistorted footage.

Generated proxy images are encoding using JPEG, and the quality of the JPEG codec is controlled with the **Quality** slider.

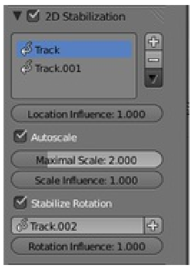
By default, all generated proxy images are storing to the <path of original footage>/BL_proxy/<clip name> folder, but this location can be set by hand using the **Proxy Custom Directory** option.

Rebuild Proxy will regenerate proxy images for all sizes set above and regenerate all timecodes which can be used later.

Use Timecode Index can (and better be used) for movie files. Basically, timecode makes frame search faster and more accurate. Depending on your camera and codec, different timecodes can give better result.

Proxy Render Size defines which proxy image resolution is used for display. If **Render Undistorted** is set, then images created from undistorted frames are used. If there's no generated proxies, render size is set to "No proxy, full render", and render undistorted is enabled, undistortion will happen automatically on frame draw.

Tools available in reconstruction mode



Location Influence: 1.000
☑ Autoscale
Maximal Scale: 2.000
Scale Influence: 1.000
☑ Stabilize Rotation
Track.002
Rotation Influence: 1.000

Proxy / 2D Stabilization Panel in clip editor

There's one extra panel which is available in reconstruction mode - 2D stabilization panel.

This panel is used to define data used for 2D stabilization of the shot. Several options are available in this panel.

First of all is the list of tracks to be used to compensate for camera jumps, or location. It works in the following way: it gets tracks from the list of tracks used for location stabilization and finds the median point of all these tracks on the first frame. On each frame, the algorithm makes this point have the same position in screen coordinates by moving the whole frame. In some cases it's not necessary to fully compensate camera jumps and **Location Influence** can be used in such cases.

The camera can also have rotated a bit, adding some tilt to the footage. There's the **Stabilize Rotation** option to compensate for this tilt. A single extra track needs to be set for this, and it works in the following way. On first frame of the movie, this track is connected with the median point of the tracks from list above and angle between horizon and this segment is ket constant through the whole footage. The amount of rotation applied to the footage can be controlled by **Rotation Influence**.

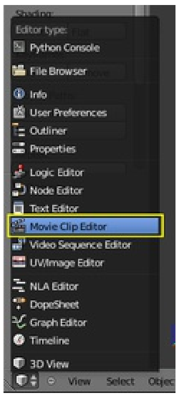
If the camera jumps a lot, there'll be noticeable black areas near image boundaries. To get rid of these black holes, there's the **Autoscale** option, which finds smallest scale factor which, when applied to the footage, would eliminate all the black holes near the image boundaries. There's an option to control the maximal scale factor, (**Maximal Scale**), and the amount of scale applied to the footage (**Scale Influence**).

Motion Tracking Tools and Properties

Movie Clip Editor

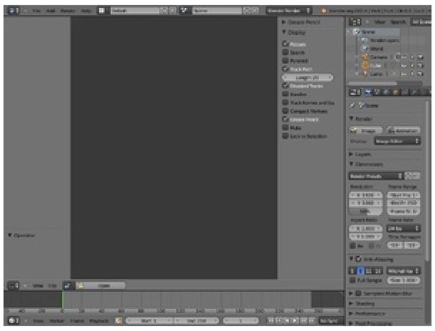
Almost all motion tracking tools are concentrated in the Movie Clip Editor. Currently it doesn't have any tools which aren't related to motion tracking, but in the future it may be expanded to be used for masking, so that's why it's given this more abstract name, rather than motion tracking.

This editor can be found in the list of editor types.



Editor type menu

When you switch to Movie Clip Editor window, the interface changes in the following way.

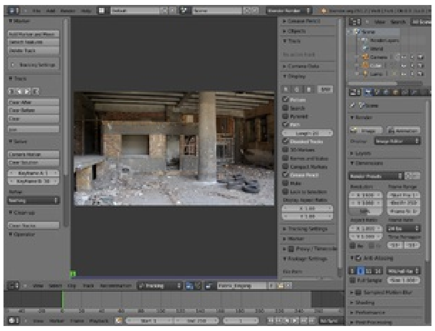

Movie Clip Editor interface

The next logical step to open a new video clip to start working with. There are several ways to to this:

- Use Open button from movie editor header
- Use Clip » Open menu
- Use AltO> shortcut

Both movie files and image sequences can be used in the clip editor. If you're using an image sequence there's one limitation on naming of files: the numbers at the end of the image name should be increasing continuously.

So, when a movie clip is loaded into the clip editor, extra panels are displayed in the interface.

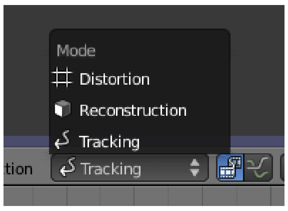

Movie Clip Editor with opened clip

There are plenty of new tools on the screen and here's short description of all of them.

First of all, it should be mentioned that the camera solver consists of three quite separate steps:

- 2D tracking of footage
- Camera intrinsics (focal length, distortion coefficients) specification/estimation/calibration
- Solving camera, scene orientation, scene reconstruction

Tools in the clip editor are split depending on which step they're used in, so the interface isn't cluttered up with scene orientation tools when only 2D tracking can be done. The currently displayed tool category can be changed using the Mode menu which is in the editor header.



Movie Clip Editor mode menu

But almost all operators can be called from menus, so it's not necessary to change the mode every time you want to use a tool which is associated with a different editor mode.

In tracking mode only tools which are related to tracking and camera solving are displayed. Camera solving tools are included here because it's after solving you'll most probably want to re-track existing tracks or place new tracks to make solving more accurate.

Tools available in tracking mode

Marker panel

- The **Add Marker and Move** operator places a new marker at the position of the mouse (which is under the button in this case, not ideal but it's just how things work) and then it can be moved to the needed location. When it's moved to the desired position, LMB can be used to finish placing the new marker. Also, ↵ Enter and Space can be used to finish placing the marker.

  But it's faster to use Ctrl LMB to place markers directly on the footage. This shortcut will place the marker in the place you've clicked. One more feature here: until you've released the mouse button, you can adjust the marker position by moving the mouse and using the track preview widget to control how accurately the marker is placed.

- The **Detect Features** operator detects all possible features on the current frame and places markers at these features. This operator doesn't take into account other frames, so it can place markers on features which belong to moving objects, and if camera is turning away from this shot, no markers would be placed on frames after the camera moved away.

  There are several properties for this operator:

  Placement is used to control where to place markers. By default, they'll be added through the whole frame, but you can also outline some areas with interesting features with grease pencil and place markers only inside the outlined area. That's how the "Inside Grease Pencil" placement variant works. You can also outline areas of no interest (like trees, humans and so) and place markers outside of these areas. That's how the "Outside Grease Pencil" placement variant works.
  Margin controls the distance from the image boundary for created markers. If markers are placed too close to the image boundary, they'll fail to track really quickly and they should be deleted manually. To reduce the amount of manual clean-up, this parameter can be used.
  Trackability limits minimal trackability for placing markers. This value comes from the feature detection algorithm and basically it means: low values means most probably this feature would fail to track very soon, high value means it's not much such track. Amount of markers to be added can be controlled with this value.
  Distance defines the minimal distance between placed markers. It's needed to prevent markers from being placed too close to each other (such placement can confuse the camera solver).

- **Delete Track** is a quite self-explaining operator which deletes all selected tracks.

Track panel

- The first row of buttons is used to perform tracking of selected tracks (i.e. following the selected feature from frame to frame). Tracking can happen (in order of buttons):
  - Backward one frame
  - Backward along the sequence
  - Forward along the whole sequence
  - Forward one frame

  This operator depends on settings from the Tracking Settings panel, which will be described later.
  If during sequence tracking the algorithm fails to track some markers, they'll be disabled and tracking will continue for the rest of the markers. If the algorithm fails when tracking frame-by-frame, the marker is not disabled, and the most likely position of the feature on the next frame is used.

- **Clear After** deletes all tracked and keyframed markers after the current frame for all selected tracks.
- **Clear Before** deletes all tracked and keyframed markers before the current frame for all selected tracks.
- **Clear** clears all markers except the current one from all selected tracks.
- **Join** operator joins all selected tracks into one. Selected tracks shouldn't have common tracked or keyframed markers at the same frame.

Solve panel

Camera Motion operator solves the motion of camera using all tracks placed on the footage and two keyframes specified on this panel. There are some requirements:

- There should be at least 8 common tracks on the both of the selected keyframes.
- There should be noticeable parallax effects between these two keyframes.

If everything goes smoothly during the solve, the average reprojection error is reported to the information space and to the clip editor header. Reprojeciton error means the average distance between reconstructed 3D position of tracks projected back to footage and original position of tracks. Basically, reprojection error below 0.3 means accurate reprojection, 0.3-3.0 means quite nice solving which still can be used. Values above 3 means some tracks should be tracked more accurately, or that values for focal length or distortion coefficients were set incorrectly.

The **Refine** option specifies which parameters should be refined during solve. Such refining is useful when you aren't sure about some camera intrinsics, and solver should try to find the best parameter for those intrinsics. But you still have to know approximate initial values - it'll fail to find correct values if they were set completely incorrectly initially.

Cleanup Panel

This panel contains a single operator and its settings. This operator cleans up bad tracks: tracks which aren't tracked long enough or which failed to reconstruct accurately. Threshold values can be specified from sliders below the button. Also, several actions can be performed for bad tracks:

- They can simply be selected
- Bad segments of tracked sequence can be removed
- The whole track can be deleted

Clip Panel

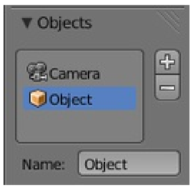
This panel currently contains the single operator Set as background which sets the clip currently being edited as the camera background for all visible 3D viewports. If there's no visible 3D viewports or the clip editor is open in full screen, nothing will happen.

Properties available in tracking mode

Grease Pencil Panel

It's a standard grease pencil panel where new grease pencil layers and frames can be controlled. There's one difference in the behavior of the grease pencil from other areas - when a new layer is created "on-demand" (when making a stroke without adding a layer before this) the default color for the layer is set to pink. This makes the stroke easy to notice on all kinds of movies.

Objects Panel



Objects Panel in clip editor

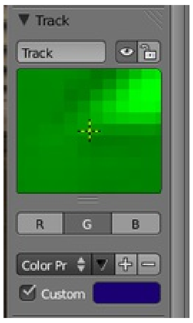
This panel contains a list of all objects which can be used for tracking, camera or object solving. By default there's only one object in this list which is used for camera solving. It can't be deleted and other objects can't be used for camera solving; all added objects are used for object tracking and solving only. These objects can be referenced from Follow Track and Object Solver constraints. Follow Track uses the camera object by default.

New objects can be added using + and the active object can be deleted with the - button. Text entry at the bottom of this panel is used to rename the active object.

If some tracks were added and tracked to the wrong object, they can be copied to another object using Track » Copy Tracks and Track » Paste Tracks.

The usage for all kind of objects (used for camera and object tracking) is the same: track features, set camera data, solve motion. Camera data is sharing between all objects and refining of camera intrinsics happens when solving camera motion only.

Track Panel



Track Panel in clip editor

First of all, track name can be changed in this panel. Track names are used for linking tracking data to other areas, like a Follow Track constraint.

The next thing which can be controlled here is the marker's enabled flag (using the button with the eye icon). If a marker is disabled, its position isn't used either by solver nor by constraints.

The button with the lock icon to the right of the button with the eye controls whether the track is locked. Locked tracks can't be edited at all. This helps to prevent accidental changes to tracks which are "finished" (tracked accurate along the whole footage).

The next widget in this panel is called "Track Preview" and it displays the content of the pattern area. This helps to check how accurately the feature is being tracked (controlling that there's no sliding off original position) and also helps to move the track back to the correct position. The track can be moved directly using this widget by mouse dragging.

If an anchor is used (the position in the image which is tracking is different from the position which is used for parenting), a preview widget will display the area around the anchor position. This configuration helps in masking some things when there's no good feature at position where the mask corner should be placed. Details of this technique will be written later.

There's small area below the preview widget which can be used to enlarge the vertical size of preview widget (the area is highlighted with two horizontal lines).

The next setting is channels control. Tracking happens in gray-scale space, so a high contrast between the feature and its background yields more accurate tracking. In such cases disabling some color channels can help.

The last thing is custom color, and the preset for it. This setting overrides the default marker color used in the clip editor and 3D viewport, and it helps to distinguish different type of features (for example, features in the background vs. foreground and so on). Color also can be used for "grouping" tracks so a whole group of tracks can be selected by color using the Select Grouped operator.

- **Marker Tip 1:** To select good points for tracking, use points in the middle of the footage timeline and track backwards and forwards from there. This will provide a greater chance of the marker and point staying in the camera shot.

Camera Data Panel

This panel contains all settings of the camera used for filming the movie which is currently being edited in the clip editor.

First of all, predefined settings can be used here. New presets can be added or unused presets can be deleted. But such settings as distortion coefficients and principal point aren't included into presets and should be filled in even if camera presets are used.

- **Focal Length** is self-explanatory; it's the focal length with which the movie was shot. It can be set in millimeters or pixels. In most cases focal length is given in millimeters, but sometimes (for example in some tutorials on the Internet) it's given in pixels. In such cases it's possible to set it directly in the known unit.
- **Sensor Width** is the width of the CCD sensor in the camera. This value can be found in camera specifications.
- **Pixel Aspect Ratio** is the pixel aspect of the CCD sensor. This value can be found in camera specifications, but can also be guessed. For example, you know that the footage should be 1920x1080, but the images themselves are 1280x1080. In this case, the pixel aspect is:

```
1920 / 1280 = 1.5
```

- **Optical Center** is the optical center of the lens used in the camera. In most cases it's equal to the image center, but it can be different in some special cases. Check camera/lens specifications in such cases. To set the optical center to the center of image, there's a Center button below the sliders.
- **Undistortion K1, K2 and K3** are coefficients used to compensate for lens distortion when the movie was shot. Currently these values can be tweaked by hand only (there are no calibration tools yet) using tools available in Distortion mode. Basically, just tweak K1 until solving is most accurate for the known focal length (but also take grid and grease pencil into account to prevent "impossible" distortion).

Display Panel

This panel contains all settings which control things displayed in the clip editor.

- **R, G, B** and **B/W** buttons at the top of this panel are used to control color channels used for frame preview and to make the whole frame gray scale. It's needed because the tracking algorithm works with gray-scale images and it's not always obvious to see which channels disabled will increase contrast of feature points and reduce noise.
- **Pattern** can be used to disable displaying of rectangles which correspond to pattern areas of tracks. In some cases it helps to make the clip view cleaner to check how good tracking is.
- **Search** can be used to disable displaying of rectangles which correspond to search areas of tracks. In some cases it helps to make the clip view cleaner to check how good tracking is. Only search areas for selected tracks will be displayed.
- **Pyramid** makes the highest pyramid level be visible. Pyramids are defined later in the Tracking Settings panel section, but basically it helps to determine how much a track is allowed to move from one frame to another.
- **Track Path** and **Length** control displaying of the paths of tracks. The ways tracks are moving can be visible looking at only one frame. It helps to determine if a track jumps from its position or not.
- **Disabled Tracks** makes it possible to hide all tracks which are disabled on the current frame. This helps to make view more clear, to see if tracking is happening accurately enough.
- **Bundles** makes sense after solving the movie clip, and it works in the following way: the solved position of each track gets projected back to the movie clip and displayed as a small point. The color of the point depends on the distance between the projected coordinate and the original coordinate: if they are close enough, the point is green, otherwise it'll be red. This helps to find tracks which weren't solved nicely and need to be tweaked.
- **Track Names and Status** displays information such as track name and status of the track (if it's keyframed, disabled, tracked or estimated). Names and status for selected tracks are displayed.
- **Compact Markers**. The way in which markers are displayed (black outline and yellow foreground color) makes tracks visible on

all kind of footage (both dark and light). But sometimes it can be annoying and this option will make the marker display more compactly - the outline is replaced by dashed black lines drawn on top of the foreground, so that marker areas are only 1px thick.
- **Grease pencil** controls if grease pencil strokes are allowed to be displayed and made.
- **Mute** changes displaying on movie frame itself with black square, It helps to find tracks which are tracked inaccurately or which weren't tracked at all.
- **Grid** (available in distortion mode only) displays a grid which is originally orthographic, but os affected by the distortion model. This grid can be used for manual calibration - distorted lines of grids are equal to straight lines in the footage.
- **Manual Calibration** (available in distortion mode only) applies the distortion model for grease pencil strokes. This option also helps to perform manual calibration. A more detailed description of this process will be added later.
- **Stable** (available in reconstruction mode only). This option makes the displayed frame be affected by the 2D stabilization settings. It's only a preview option, which doesn't actually change the footage itself.
- **Lock to Selection** makes the editor display selected tracks at the same screen position along the whole footage during playback or tracking. This option helps to control the tracking process and stop it when the track is starting to slide off or when it jumped.
- **Display Aspect Ratio** changes the aspect ratio for displaying only. It does not affect the tracking or solving process.

Tracking Settings Panel

Common options

This panel contains all settings for the 2D tracking algorithms. Depending on which algorithm is used, different settings are displayed, but there are a few that are common for all tracker settings:

Adjust Frames controls which patterns get tracked; to be more precise, the pattern from which frame is getting tracked. Here's an example which should make things clearer.

The tracker algorithm receives two images inside the search area and the position of a point to be tracked in the first image. The tracker tries to find the position of that point from the first image in the second image.

Now, this is how tracking of the sequence happens. The second image is always from a frame at which the position of marker isn't known (next tracking frame). But a different first image (instead of the one that immediately precedes the second image in the footage) can be sent to the tracker. Most commonly used combinations:

- An image created from a frame on which the track was keyframed. This configuration prevents sliding from the original position (because the position which best corresponds to the original pattern is returned by the tracker), but it can lead to small jumps and can lead to failures when the feature point is deformed due to camera motion (perspective transformation, for example). Such a configuration is used if **Adjust Frames** is set to 0.

- An image created from the current frame is sent as first image to the tracker. In this configuration the pattern is tracking between two neighboring frames. It allows dealing with cases of large transformations of the feature point but can lead to sliding from the original position, so it should be controlled. Such a configuration is used if **Adjust Frames** is set to 1.

If **Adjust Frames** is greater than 1, the behavior of tracker is: keyframes for tracks are creating every **Adjust Frames** frames, and tracking between keyframed image and next image is used.

Speed can be used to control the speed of sequence tracking. This option doesn't affect the quality of tracking; it just helps to control if tracking happens accurately. In most cases tracking happens much faster than real time, and it's difficult to notice when a track began to slide out of position. In such cases **Speed** can be set to Double or Half to add some delay between tracking two frames, so slide-off would be noticed earlier and the tracking process can be cancelled to adjust positions of tracks.

Frames Limit controls how many frames can be tracked when the Track Sequence operator is called. So, each Track Sequence operation would track maximum **Frames Limit** frames. This also helps to notice slide-off of tracks and correct them.

Margin can be used disable tracks when they become too close to the image boundary. This slider sets "too close" in pixels.

KLT tracker options

The KLT tracker is the algorithm used by default. It allows tracking most kinds of feature points and their motion. It uses pyramid tracking which works in the following way. The algorithm tracks an image larger than the defined pattern first to find the general direction of motion. Then it tracks a slightly smaller image to refine the position from the first step and make the final position more accurate. This iterates several times. The number of steps of such tracking is equal to the **Pyramid Level** option and we tell that on first step tracking happens for highest pyramid level. So Pyramid Level=1 is equal to pattern itself, and each next level doubles tracking image by 2.

The search area should be larger than the highest pyramid level and the "free space" between the search area and highest pyramid level defines how much the feature can move from one frame to another and still be tracked.

Default settings should work in most general cases, but sometimes the pyramid level should be changed. For example, when footage is blurry, adding extra pyramid levels helps to track them.

This algorithm can fail in situations where a feature point is moving in one direction and the texture around that feature point is moving in another direction.

SAD tracker options

On each step, the SAD tracker reviews the whole search area and finds the pattern on the second image which is most like the pattern which is getting tracking. This works pretty quickly, but can fail in several cases. For example, when there's another feature point which looks like the tracking feature point in the search area. In this case, SAD will tend to jump off track from one feature to another.

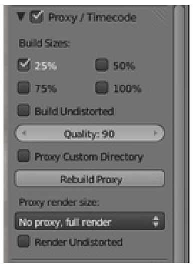
Correlation defines the threshold value for correlation between two patterns which is still considered successful tracking. 0 means there's no correlation at all, 1 means correlation is full.

There's one limitation: currently: it works for features of size 16x16 pixels only.

Marker Panel

This panel contains numerical settings for marker position, pattern and search area dimensions, and offset of anchor point from pattern center. All sliders are self-explanatory.

Proxy / Timecode Panel



Proxy / Timecode Panel in clip editor

This panel contains options used for image proxies and timecodes for movies.

Proxy allows displaying images with lower resolution in the clip editor. This can be helpful in cases when tracking of 4K footage is happening on a machine with a small amount of RAM.

The first four options are used to define which resolutions of proxy images should be built. Currently it's possible to build images 25%, 50%, 75% and 100% of the original image size. Proxy size of 100% can be used for movies which contain broken frames which can't be decoded.

Build Undistorted means that the proxy builder also creates images from undistorted original images for the sizes set above. This helps provide faster playback of undistorted footage.

Generated proxy images are encoding using JPEG, and the quality of the JPEG codec is controlled with the **Quality** slider.

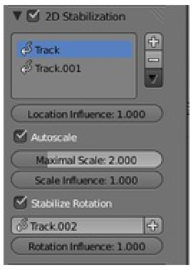
By default, all generated proxy images are storing to the <path of original footage>/BL_proxy/<clip name> folder, but this location can be set by hand using the **Proxy Custom Directory** option.

Rebuild Proxy will regenerate proxy images for all sizes set above and regenerate all timecodes which can be used later.

Use Timecode Index can (and better be used) for movie files. Basically, timecode makes frame search faster and more accurate. Depending on your camera and codec, different timecodes can give better result.

Proxy Render Size defines which proxy image resolution is used for display. If **Render Undistorted** is set, then images created from undistorted frames are used. If there's no generated proxies, render size is set to "No proxy, full render", and render undistorted is enabled, undistortion will happen automatically on frame draw.

Tools available in reconstruction mode



Proxy / 2D Stabilization Panel in clip editor

There's one extra panel which is available in reconstruction mode - 2D stabilization panel.

This panel is used to define data used for 2D stabilization of the shot. Several options are available in this panel.

First of all is the list of tracks to be used to compensate for camera jumps, or location. It works in the following way: it gets tracks from the list of tracks used for location stabilization and finds the median point of all these tracks on the first frame. On each frame, the algorithm makes this point have the same position in screen coordinates by moving the whole frame. In some cases it's not necessary to fully compensate camera jumps and **Location Influence** can be used in such cases.

The camera can also have rotated a bit, adding some tilt to the footage. There's the **Stabilize Rotation** option to compensate for this tilt. A single extra track needs to be set for this, and it works in the following way. On first frame of the movie, this track is connected with the median point of the tracks from list above and angle between horizon and this segment is ket constant through the whole footage. The amount of rotation applied to the footage can be controlled by **Rotation Influence**.

If the camera jumps a lot, there'll be noticeable black areas near image boundaries. To get rid of these black holes, there's the **Autoscale** option, which finds smallest scale factor which, when applied to the footage, would eliminate all the black holes near the image boundaries. There's an option to control the maximal scale factor, (**Maximal Scale**), and the amount of scale applied to the footage (**Scale Influence**).

Mask

Introduction

Mask is a grayscale raster image which is created from vector image. Artists interacts with vector image to create mask with needed shape and then it automatically gets rasterized when using in Node Editor or wherever else.

User interacts with masks in the following editors:

- Movie Clip Editor
- Image Editor
- Node Editor
- Sequencer
- Dope Sheet

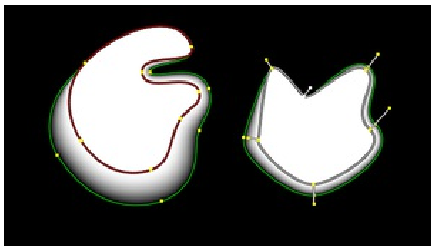
Movie Clip Editor and Image editor are used to edit masks, Compositor and Sequencer are just using already created mask, Dope Sheet for animating masks.

Previously there was no simple workflow for masks in Blender, compositing a rendered scene with real footage was possible, but when it came to masking out objects, defining areas of influence and other scenarios, the workflow was cumbersome. Masks are now natively supported, which allow you to draw masks using splines, and then have them rasterized for use in the compositor or sequencer.

This features consists of a few different parts:

- Mask datablock containing multiple mask layers and splines.
- Mask editing in the image and movie clip editor space using various tools.
- Animation of masks with keyframes, drivers and tracking data.
- Compositing node and sequencer strip to use mask.

Mask Datablock: Points, Splines and Layers


Two splines with feathering.

A **Point** is the most low-level entity used to define mask. It's a simple point with it a coordinate, handles and set of feather points. Points can be parented to markers from motion tracking.

Sets of points define a **Spline**. Currently only Bezier splines are supported. They create a smooth curve from the first to the last point in the spline. Splines by default will create a filled area, but can also create non-closed curves with a thickness to mask out objects such as wires or hair.

One or several splines can belong to the same **Layer**. Splines belonging to the same layer can be animated together, for example by an item from motion tracker footage. By creating overlapping splines holes can be created, and it's the layer membership that defines which splines interact to create holes.

Mask datablocks are the most high-level entity used for masking purposes. They can be reused in different places, and hold global parameters for all the entities they consist of.

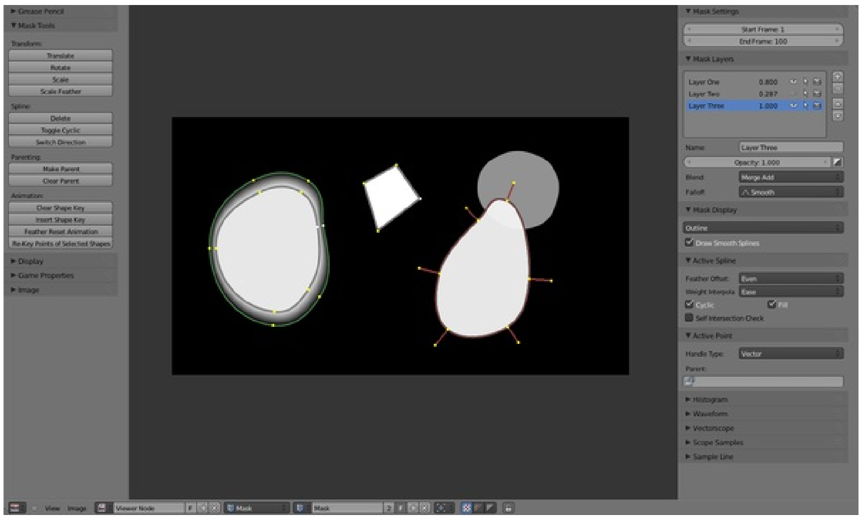
Understanding Layers

The purpose of mask layers can be explained with an example. Suppose there are two unwanted people in the footage, and one of them goes from left to right, and the other in the opposite direction. Two mask layers can then be used to mask them separately using a single mask datablock. At the point of intersection of these shapes they will be added together rather than creating a hole, as would happen if they were on the same layer. If the motion is simple enough, a single motion tracked point can be used to drive the location of the entire mask layer. Each mask layer can consist of multiple splines to fit more complex shapes.

Editing Masks

Masks can be created in the image and movie clip editors, by changing the mode from View to Mask in the header. This will add various tools and properties to the editor panels, while hiding others that are not needed for interacting with masks. The tools and panels available to edit masks are the same in both editors, with the exception that linking masks to motion tracking data is only possible in the movie clip editor.

Once set to Mask mode, a Mask datablock can be added. Any image, movie clip, render or compositing result can be used as a backdrop to draw masks over. To get interactive feedback on the resulting mask, a Mask node can be connected directly to a Viewer node in the compositor, which will then keep updating the compositing result while editing.


Mask editing in image editor

Control Points

Editing of mask splines happens in a way similar to editing bezier curves or paths in GIMP or other curve editors: control points are added to define the spline itself, and handles of different types are used to create smooth bends. This makes it possible to define a mask with few points to easily follow an object in footage.

- Ctrl + LMB is used to place new control points and define handle orientations (click to place control point, click followed with slide to place new control point and set smoothness for it).
- Alt + C: to close the mask by joining the last control point to the first.
- Existing control points can be translated, scaled and rotated with the usual G, S, R shortcuts.
- X or Delete removes control points.

Selection

The usual selection and hide/reveal tools are available:

- A: toggle select all
- B, C: border and circle Select
- Ctrl + L select linked from selection, L: select linked with mouse
- Ctrl + Alt + LMB: lasso select
- H hide selected, ⇧ Shift + H hide unselected, Alt + H reveal

Curve Handles

- Alt + C: cycle toggle spline, to create a close curve or open it again
- V: set handle type for selected spline points
- Ctrl + N: make normals (handle directions) consistent
- Switch Direction handle directions in/out.

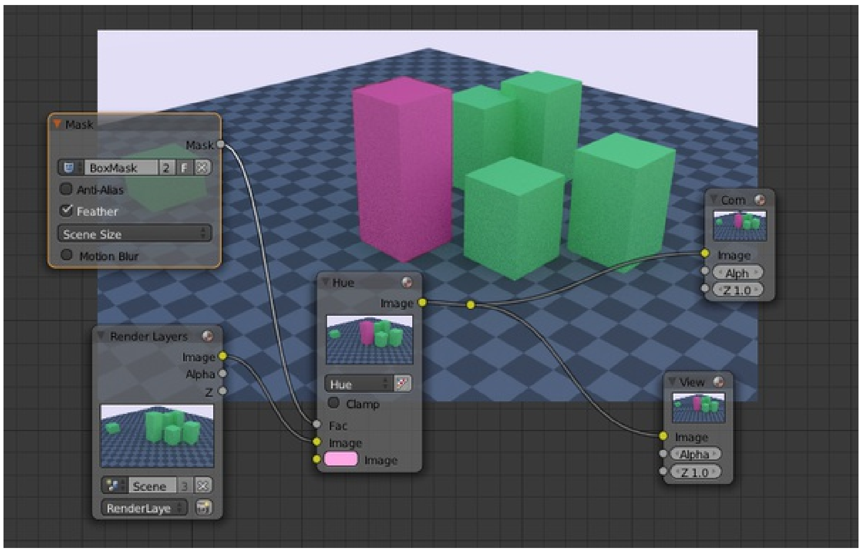
Feather

It's possible to control feather of mask, including a way to define non-linear feather. Linear feather is controlled by a slider, non-linear feather is controlled in the same curve-based way to define feather falloff.

- ⇧ Shift + LMB is used to define a feathering outline curve. To create an initial feather, sliding from a spline control point outside or inside will create and position feather points. After this ⇧ Shift + LMB will insert new feather point and mouse sliding can be used to move them around.
- Alt + S will scale the feather size.

Using Masks

Masks have many purposes. They can be used in a motion tracking workflow to mask out, or influence a particular object in the footage. They can be used for manual rotoscoping to pull a particular object out of the footage, or as a rough matte for green screen keying. Masks are independent from a particular image of movie clip, and so they can just as well be used for creating motion graphics or other effects in the compositor.



Using the Mask node to isolate an object in compositing

Compositing Node

In the compositing nodes the Mask input node can be used to select a mask datablock, with as output the raster mask image. This image can be used with other nodes, for example to Invert, Multiply or Mix, or use as a factor input. The node options are:

Anti-Alias
: Create smooth mask edges rather than hard ones.

Feather
: Use or ignore feather points defined for splines.

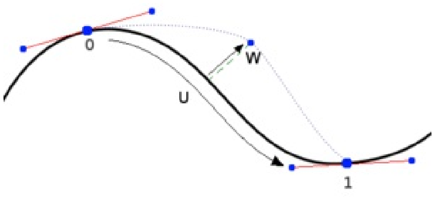
Size
: Scene Size will give an image the size of the render resolution for the scene, scaling along when rendering with different resolutions. Fixed gives a fixed size in pixels. Fixed/Scene gives a size in pixels that still scales along when changing the render resolution percentage in the scene.

Motion Blur
: For animated masks, creating a motion blurred mask from the surrounding frames, with a given number of samples (higher gives better quality), and a camera shutter time in seconds.

Sequencer Strip

In the sequencer a Mask strip can be added, which generates a mask image. This works similar to the compositing node but without the options available for finer control. The mask image is always generated at the render resolution, scaling along with different proxy levels.

S-Curves



S-Curve

The curve type used for creating mask splines is almost a Bezier curve, but with some differences. The curve needed to support feathering in a way that stuck to the curve as you edited it, for ease of editing an animation. We call these S-Curves.

Besides the handles, every control point also has points that define the feather between the current point and the next point on the spline. Each feather point is stored in UV space, where U means position across spline segment, and V means distance between main spline and feather points.

This allows for deforming the main spline in almost any way, and the feather will be updated automatically to reflect that change. For example if there's just rotation of the spline, feather would stay completely unchanged. If one point's feather is moved, the other feathers will be automatically stretched uniformly along that segment and the overall shape will be almost the same as artists would want it to be.

Masking Tips

1. Analyze the sequence. Try to figure out which parts of the object are overlapping, moves in different directions (arms, legs, body of walking man; protruding elements and body of a car (eg: mirrors, antenas, lights on the roof); ears and nose when head turns). For each of those overlapped objects you should create separate mask. Obvious masks also should be overlapped, not end-to-end (only just touching).

Its a good idea is to create separate masks for parts with different kinds of motion. For example, if a man stays still and shakes his head actively, you can create separate masks for head and figure. But someone moves across the screen in a straight line and hold his head straight, you'll need separate masks for arms and legs, and you can create single mask for head and body.

It's important to create masks namely for objects, not for spaces between them. For example, if a man is waving a flag held with both hands – Then the overall shape the hands, body and flagstaff would be much more consistent and easy for masking, than it would be if the whole body's silhouette and spaces between arms and body.

Never try masking out a group of moving people with a single mask. The same thing applies for clenching fist. In this case you should use separate masks for each knuckle. Also for complex shapes, use several masks with just one "working" side (see below)

2. Always use tracking where possible. If the shape of the object changing slightly, and you have translating, rotation or zoom in the sequence, you can track the motion and attach mask vertices's to the track. In this way you'll get general motion and you'll need create much less keyframes for compensate inaccuracy of tracking.

3. Find the frame where you can see whole object. Start masking from this frame. Try to use as few mask points as possible for the objects outline. Otherwise it could be hard to animate.

Next, to avoid jittering and sudden movements of mask, animate the mask in several passes. Firstly create basic keyframes and then add keyframes where its necessary.

4. First pass. Find the key moments – beginning of objects moving, stopping, changing speed. Create certain amount of keyframes accordingly.

While animating a mask its easy to forget the origins of the mask points – so for eg: what starts as the point of an elbow may end on the wrist by the end of the animation. – This should be avoided because the areas in-between the keyframes tend to look bad and you end up adding more keyframes to compensate for it.

5. Second pass. Check how mask interpolates between keyframes and how it follows the movement of object. Add keyframes where the mask has the most significant offset from the object. Repeat this 3 or four times.

Once you have the bare minimum number of keyframes needed, the masks animation will look smooth so you'll be able to adjust mask's shape with less effort.

Now you can enable motion blur on the mask node, adjusting individually for each shot. In Blender I usually use Shutter = 0.25-0.35.

Render engines

- Blender Internal
- Cycles

Rendering

Rendering is the final process of CG (short of post processing, of course) and is the phase in which a 2D image corresponding to your 3D scene is finally created. Rendering is a CPU-intensive process. You can render an image on your computer, or use a render farm which is a network of PCs that each work on a different section of the image or on different frames. This section provides a full explanation of the features in Blender related to the process of producing your image or animation.

After you have set up the materials, textures, lighting, and the camera, you can begin rendering. It is unlikely that you will get it right on the first render, so be prepared to do many test renderings. This section describes the options and settings for the rendering process that will result in the desired image quality.

Blender has in internal render engine that it uses. This is a fast renderer, and can produce nice results if fine tuned. There are several other external renderers that can be loaded, which offer more advanced rendering tools.

We know that around the world, our users have PCs of widely varying power. Rendering is *the* process in CG that can chew up CPU and disk space like there's no tomorrow. Especially in corporate environments, it is easy to fill up terabyte servers by uploading ten hour-long DV tapes and doing some editing. So there are lots of options to try to shoehorn a big job into a small PC by providing you with multiple sets of options that chunk up the work as best we can, while still preserving image integrity.

This page discusses the main options found on the Render panel, and subsequent pages explain more.

Overview

The rendering of the current scene is performed by pressing the big Image button in the Render panel, or by pressing F12 (you can define how the rendered image is displayed on-screen in the Render Output Options). See also the Render Window.

A movie is produced by pressing the big Animation button. The result of a rendering is kept in a buffer and shown in its own window. It can be saved by pressing F3 or via the File->Save Image menu using the output option in the Output panel. Animations are saved according to the format specified, usually as a series of frames in the output directory. See Output Options and Animations.

The image is rendered according to the dimensions defined in the Dimensions Panel.

Workflow

In general, the process for rendering is:

1. Create all the objects in the scene
2. Light the scene
3. Position the Camera
4. Render a test image at 25% or so without oversampling or ray tracing etc., so that it is very fast and does not slow you down
5. Set and adjust the materials/textures and lighting
6. Iterate the above steps until satisfied with the quality level
7. Render progressively higher-quality full-size images, making small refinements and using more compute time
8. Save your images

Distributed Render Farm

There are several levels of CPU allocation that you can use to decrease overall render time by applying more brainpower to the task.

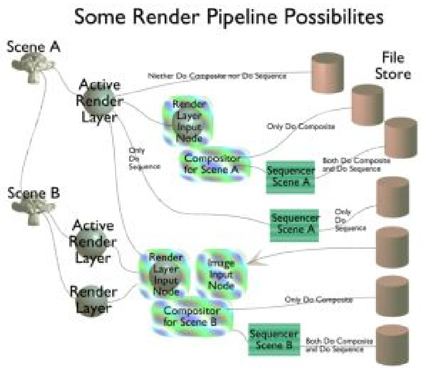
First, if you have a multi-core CPU, you can increase the number of threads, and Blender will use that number of CPUs to compute the render.

Second, if you have a local area network with available PCs, you can split the work up by frames. For example, if you want to render a 200-frame animation, and have 5 PCs of roughly equal processing power, you can allocate PC#1 to produce frames 1-40, PC#2 to frames 41-80, and so on. If one PC is slower than the others, simply allocate fewer frames to that PC. To do LAN renders, map the folder containing the .blend file (in which you should have packed your external data, like the textures, …) as a shareable drive. Start Blender on each PC and open the .blend file. Change the Start and End frame counts on that PC, but do not save the .blend file. Start Rendering. If you use relative paths for your output pathspec, the rendered frames will be placed on the host PC.

Third, you can do WAN rendering, which is where you email or fileshare or Verse-share the .blend file (with packed data!) across the Internet, and use anyone's PC to perform some of the rendering. They would in turn email you the finished frames as they are done. If you have reliable friends, this is a way for you to work together.

Fourth, you can use a render farm service. These services, like BURP, are run by an organization. You email them your file, and then they distribute it out across their PCs for rendering. BURP is mentioned because it is free, and is a service that uses fellow Blender users' PCs with a BOINC-type of background processing. Other services are paid subscriptions or pay-as-you-go services.

Render Workbench Integration



Blender has three independent rendering workbenches which flow the image processing in a pipeline from one to the other in order:

- Rendering Engine
- Compositor
- Sequencer

You can use each one of these independently, or in a linked workflow. For example, you can use the Sequencer by itself to do post-processing on a video stream. You can use the Compositor by itself to perform some color adjustment on an image. You can render the scene via the active Render Layer, and save that image directly, with the scene image computed in accordance with the active render layer, without using the Compositor or Sequencer. These possibilities are shown in the top part of the image to the right.

You can also link scenes and renders in Blender as shown, either directly or through intermediate file storage. Each scene can have multiple render layers, and each Render Layer is mixed inside the Compositor. The active render layer is the render layer that is displayed and checked active. If the displayed render layer is not checked active/enabled, then the next checked render layer in the list is used to compute the image. The image is displayed as the final render if Compositing and Sequencer are NOT enabled.

If Compositing is enabled, the render layers are fed into the Compositor. The nodes manipulate the image and send it to the Composite output, where it can be saved, or, if *Do Sequence* is on, it is sent to the Sequencer.

If Sequencer is enabled, the result from the compositor (if Do Composite is enabled) or the active Render layer (if Do Composite is not enabled) is fed into the Scene strip in the Sequencer. There, it is manipulated according to the VSE settings, and finally delivered as the image for that scene.

Things get a little more complicated when a .blend file has multiple scenes, for example Scene A and Scene B. In Scene B, if Compositing is enabled, the Render Layer node in Scene B's compositor can pull in a Render Layer from Scene A. Note that this image will not be the post-processed one. If you want to pull in the composited and/or sequenced result from Scene A, you will have to render Scene A out to a file using Scene A's compositor and/or sequencer, and then use the Image input node in Scene B's compositor to pull it in.

The bottom part of the possibilities graphic shows the ultimate blender: post-processed images and a dynamic component render layer from Scene A are mixed with two render layers from Scene B in the compositor, then sequenced and finally saved for your viewing enjoyment.

These examples are only a small part of the possibilities in using Blender. Please read on to learn about all the options, and then exercise your creativity in developing your own unique workflow.

The Render Settings Panel

The Render tab contains all of the options for the internal render engine, or an external one, if selected.

Render

Here you can activate the rendering process, by rendering a Still Image or an Animation.

You can also select where the image is rendered to. This are described on the Render Display page.

Layers

The Layers menu contains options for rendering in Layers and Passes

Dimensions

This menu has settings for the size of the rendered images (see Output Options), and options for rendering sequences (see Animations)).

Anti-Aliasing

Antialiasing is important for producing high quality renders that do not have "jaggies" or stair-stepped pixel artifacts.

Motion Blur

Motion Blur is an important effect in rendering moving images. It prevents the animation from appearing unrealistic and stuttery, as in stop-motion, where each frame is a perfect still image.

Shading

These are options for controlling what shading effects are calculated in the render. Deselecting them disables them.

- Textures
- Shadows
- Subsurface Scattering
- Environment Maps
- Ray Tracing
- Color Management

  Uses a linear workflow when enabled

- Alpha

  Set how transparent pixels are rendered.

Output

Set where images are rendered to and what file type. See Output Options.

Performance

Control the way the renderer performs with respect to the computer's memory and processor. See Performance.

Post Processing

Control effects that are applied after the image has been rendered. If you are using the Compositor or Sequencer, you can tell Blender to process those effects instead of directly rendering the scene.

Fields are used when Rendering for Video.

Dithering is method of blurring pixels.

You can also enable Edge Rendering to create sketch-like or toon-like effects.

Stamp

Stamping inserts text over the rendered images, as well as stamps meta-data into image formats that support it (PNG, JPEG and EXR).

Bake

Render Baking is a process that creates texture files that hold desired rendered effects, like lighting, shadows, or color information. This is useful for working with real-time graphics that benefit from not having to calculate shading when not necessary.

The Camera

A Camera is an object that provides a means of rendering images from Blender. It defines which portions of a scene is visible at render time. Scenes can have multiples cameras, but *must* contain, at minimum, one camera in order to generate any images.

Add a new camera

Mode: Object mode

Hotkey: ⇧ ShiftA to add new.

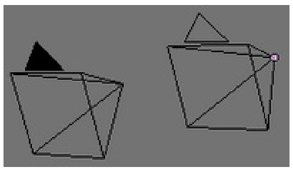
Menu: Add » Camera

In Object mode simply press ⇧ ShiftA and in the popup menu, choose Add » Camera.

Change active camera

Mode: Object mode

Hotkey: Ctrl0 NumPad


Active camera (left one).

The *active* camera is the camera that is currently being used for rendering and camera view (0 NumPad). Select the camera you would like to make active and press Ctrl0 NumPad (by doing so, you also switch the view to camera view). In order to render, each scene **must** have a camera.

The active camera is the one with the filled “up” triangle on top seen in the 3D viewport, for example, the left camera in (*Active camera (left one)*).

The active camera, as well as the layers, can be specific to a given view, or global (locked) to the whole scene – see this part of the 3D view options page.

Switching Multiple Cameras

Switching Cameras.

To change cameras mid-animation, you need to use markers.

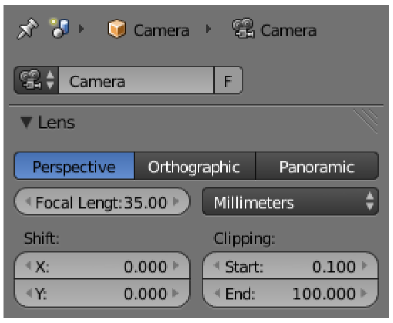
See details about switching cameras here

Camera Settings

Mode: Object mode

Panel: Camera

Cameras are invisible in renders, so they don’t have any material or texture settings. However, they do have Object and Editing setting panels available which are displayed when a camera is the selected (active!) object.

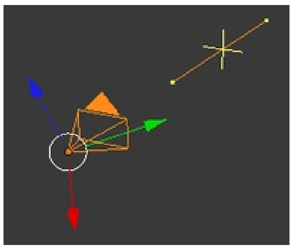


Camera Lens panel.

Lens

- Perspective / Orthographic / Panoramic

  Select what projection type to use. Perspective is the default and makes objects further away appear smaller while Orthographic maintains the exact measures of objects. A Perspective projection is more similar to what an image obtained with a real camera would look like while an Orthographic projection is a more technical view, best for blueprints, but worst to convey distances between objects.
  To configure these projections, see this page on vanishing points and isometric view.
  Panoramic renders the scene with a cylindrical projection.


A camera with the clipping limits and focal point visible.

- Focal Length

  Available in Perspective and Panoramic camera types, represents the lens focal length, represented in degrees or millimeters. When Orthographic mode is selected, the Focal Length setting changes to the Orthographic Scale setting. This setting determines the size of the camera's visible area.

- Shift X/Y

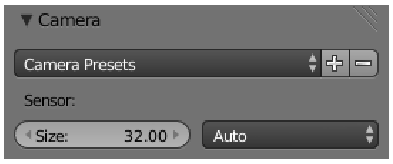
  Shifts the camera viewport. Note that most of the time, this setting should not be used to adjust the camera position, as the Shift setting is relative to the actual camera position, which will not be changed.

- Clipping Start/End

  Sets the clipping limits. Only objects within the limits are rendered. If Limits in the Display panel is enabled, the clip bounds will be visible as two yellow connected dots on the camera line of sight.

Note
The 3D View window contains settings similar to the camera, such as Orthographic/Perspective and Clip Start/Clip End. These settings have no effect on the camera rendering, and only change the view settings when *not* in Camera view. These settings are accessed through the View menu of the 3D View. See the 3D view options page for more details.

Camera Presets

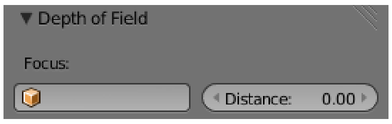


Camera Presets panel.

ToDo

- Camera Presets
- Sensor

Depth of Field



Camera Display panel

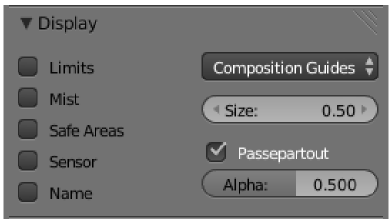
- Depth of Field object

  When using Depth of Field, the linked object will determine the focal point. Linking an object will deactivate the distance parameter.

- Distance

  Distance to the focal point. It is shown as a yellow cross on the camera line of sight. Limits must be enabled to see the cross. It is used in combination with the Defocus Compositing Node.

Display



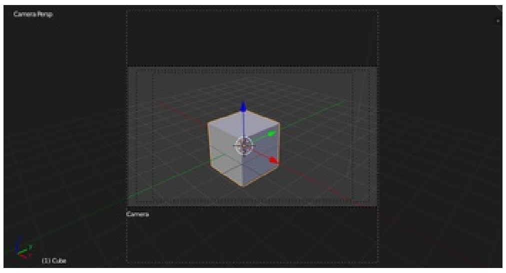
Camera Display panel

- Limits

  Toggles viewing of the limits on and off.

- Mist

  Toggles viewing of the mist limits on and off. The limits are shown as two connected white dots on the camera line of sight. The mist limits and other options are set in the World panel, in the Mist section.


Camera view displaying safe areas, sensor and name

- Safe Areas

  When this is enabled, extra dotted frames are drawn when in camera view, delimiting the area considered as “safe” for important things.

- Sensor

  Displays a dotted frame in camera view.

- Name

  Toggle name display on and off in camera view.

- Size

  Size of the camera icon in the 3D view. This setting has no effect on the render output of a camera, and is only a cosmetic setting. The camera icon can also be scaled using the standard Scale S transform key.

- Passepartout, Alpha

  This mode darkens the area outside of the camera’s field of view, based on the Alpha setting.

Composition Guides

Composition Guides are available from the drop-down menu, which can help when framing a shot. There are 8 types of guides available:

- Center

  Adds lines dividing the frame in half vertically and horizontally.

- Center Diagonal

  Adds lines connecting opposite corners.

- Thirds

  Adds lines dividing the frame in thirds vertically and horizontally.

- Golden

  Divides the width and height into Golden proportions (About 0.618 of the size from all sides of the frame).

- Golden Triangle A

  Draws a diagonal line from the lower-left to upper-right corners, then adds perpendicular lines that pass through the top left and bottom right corners.

- Golden Triangle B

  Same as A, but with the opposite corners.

- Harmonious Triangle A

  Draws a diagonal line from the lower-left to upper-right corners, then lines from the top left and bottom right corners to 0.618 the lengths of the opposite side.

- Harmonious Triangle B

  Same as A, but with the opposite corners.

Camera Navigation

Here you will find some handy ways to navigate and position your camera in your scene.

Note
Remember that the active “camera” might be any kind of object. So these actions can be used e.g. to position and aim a lamp…

Move active camera to view

Mode: Object mode

Hotkey: CtrlAlt0 NumPad

This feature allows you to position and orient the active camera to match your current viewport.

Select a camera and then move around in the 3D view to a desired position and direction for your camera (so that you're seeing what you want the camera to see). Now press CtrlAlt0 NumPad and your selected camera positions itself to match the view, and switches to camera view.

Camera View Positioning

By enabling Lock Camera to View in the View menu of the View Properties panel, while in camera view, you can navigate the 3d viewport as usual, while remaining in camera view. Controls are exactly the same as when normally moving in 3d.

Roll, Pan, Dolly, and Track

To perform these camera moves, the camera must first be *selected*, so that it becomes the active object (while viewing through it, you can RMB -click on the solid rectangular edges to select it). The following actions also assume that you are in camera view (0 NumPad)! Having done so, you can now manipulate the camera using the same commands that are used to manipulate any object:

Roll: Press R to enter object rotation mode. The default will be to rotate the camera in its local Z-axis (the axis orthogonal to the camera view), which is the definition of a camera “roll”.

Vertical Pan or Pitch: This is just a rotation along the local X-axis. Press R to enter object rotation mode, then X twice (the first press selects the *global* axis – pressing the same letter a second time selects the *local* axis – this works with any axis; see the axis locking page).

Horizontal Pan or Yaw: This corresponds to a rotation around the camera’s local Y axis… Yes, that’s it, press R, and then Y twice!

Dolly: To dolly the camera, press G then MMB (or Z twice).

Sideways Tracking: Press G and move the mouse (you can use X twice or Y to get pure-horizontal or pure-vertical sideways tracking).

Aiming the camera in Flymode

When you are in Camera view, the fly mode actually moves your active camera…

[video link]

Perspective in Render

When you press F12 and get your render, you see an image as seen through the camera's "perspective". Like how you can *view* your model in 3D View from the top, front, side, or user perspective, you can *render* your object from different perspectives. This perspective takes into account the lens size, type, and offset in giving you that picture. Each perspective uses a different number of vanishing points.

If you look at a 3D image of a cube, you will see three kinds of edges: vertical, horizontal, and depth. If all of the vertical edges are exactly parallel, there is no vanishing point for them. If however, they are not parallel, if you extended them by continuing them with a ruler, they would at some point intersect. That point is called the vanishing point.

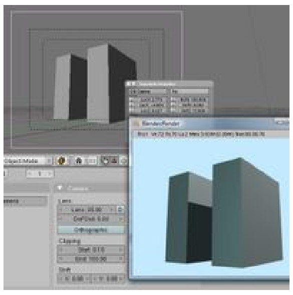
For special purposes, different kinds of render cameras can be set up to give you different perspectives. For reasons discussed below, you may wish to limit the number of vanishing points, especially for architectural purposes. Architects and drafting people are responsible for rendering the object or building with true dimensions and true relative proportions.

If you look at that example render, the building looks all sorts of distorted, like it had been made of mud and was collapsing. If you told a builder to build that, you would end up with a building that actually had leaning walls and rooms that were narrower at the top.

Way back in the old Greek days, when they started building tall columns, they built them thicker at the top than at the bottom, so that when viewed looking up, the two sides would look straight up and down. Then they even started narrowing the columns at the top to give the illusion that the building was taller and would look higher. During the Renaissance, the concept of using vanishing points in art evolved. Blender offers a few tricks of its own to let you do the same.

Note
To follow sections below you will need to know how to adjust Camera Settings

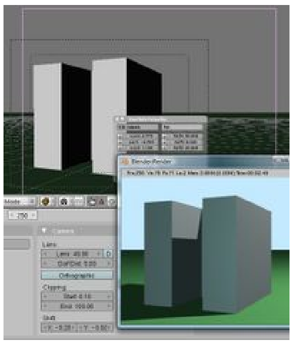
Three Point Rendering


Normal Three Point Render
When looking at or rendering a picture of a high building from ground level off to one side, and aiming up, using the normal 35mm camera, you get 3 point perspective. If you laid a ruler along the vertical lines, you would see that they converge to a point above the building.

The horizontal lines are converging off to one side (the left in this example), and depth (receding) lines are converging to a different third point (somewhere off to the lower right in this example). Hence the name 3-point rendering - there are three vanishing points.

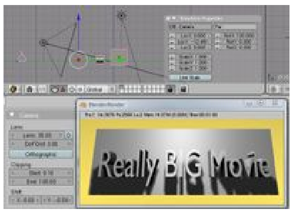
This is reality, and there is nothing wrong with that. When you next step outside and look at a tall building, this is what you actually see. However, your mind knows that the building is square, and can adjust your perception of the building so that you are not scared that the building is going to fall over.

Two Point Rendering


Two Point Vertical Render
Normal architectural rendering is called two point rendering; when vertical lines are parallel, and horizontals, if followed out to the side, converge on one point, and receding or depth lines converge to a second point.

Architects often like this Two-Point rendering, so that the sides of their buildings are completely vertical and don't appear to be falling inward. This is also quite nice for compositions and schematics, given that the lines of the paper you print on and the screen you view with are also straight.

Previously to get a 2-point perspective, you had to aim the camera level to the horizon, however this resulted in the top half of the building being cut off and the horizon being in the exact middle, which looks very boring. Architectural photographers use 'shift lenses' to solve this problem. Shift lenses shift the image to another place on the film.


Two Point Horizontal Render
This technique works well for high buildings as well as for normal sized objects.

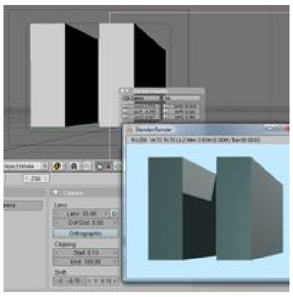
Most of the time, the two vanishing points are horizontal and depth lines, with the vertical lines parallel. However, some titles are done with the horizontal lines parallel, and the vertical and depth lines having the vanishing point. This dramatizes and exaggerates the massiveness and height of the title.

To get this effect, position the camera at ground level, centered, angle the camera upward, and shift the render passpartout down. In the example, the camera is rotated 30 degrees upward, at ground level with the title. A bright key light with a short falloff provides dramatic lighting that is bright in the middle and falls off toward the sides, further enhancing the depth.

To achieve 2-point rendering:

- Use a short wide angle lens camera, say with a Lens Size of 10 mm placed close to the building, or a long lens farther away from the building. These differences affect the depth of the building render, with longer lenses making the building appear thinner and less dramatic or distorted. The example uses a 40mm lens.
- Position the camera off to one side of the object, vertically halfway up the building to minimize distortion of the vertical building edges. You may alter this vertical (Z value) position to be slightly higher than ground level or higher than the top (if you want to see the top of the object or building). To show the front bottom corner of the building jutting out, raise up the camera.
- Angle the camera to be looking away from the building and directly level at the horizon - not pointed up or down (note the 20 degree Z angle in the example). This should make the vertical lines parallel. The more the camera looks at the object, the closer the vanishing point for the horizontal lines, and perceived depth will increase as that vanishing point gets closer as well.
- You may have to angle the camera slightly down (just 1 degree or so) so that vertical lines appear vertically up and down, both near and far. If the lines are curved, use a longer lens. With your 3D View set to Camera view, use the passpartout or pixels on your monitor to determine vertical.
- Move the camera toward/away from the object until it appears near a corner of the render and is the right size.
- Adjust the Shift: X and Y settings until your object is positioned properly.

One Point Rendering


One Point Render

One point rendering is where vertical and horizonal lines are parallel, and depth lines converge at one point. Architects really like these renders, since the front-facing faces are true and square, and the building recedes off into the distance so that it looks like it has some depth.

If the camera is placed at ground level, even with the bottom of the building, it really looks dramatic but orderly in a weird sort of way. Title graphics are sometimes rendered this way.

To get 1-point (1pt) renders

- To get more dramatic depth lines, use a short wide angle lens camera, say with a Lens Size of 10 mm, very close to the building. For a more normal appearance, stick with the 35mm lens.
- Position the camera off to one side of the object, slightly higher than the top (if you want to see the top of the object) or at ground level (the example image has the camera almost at ground level). If you position the camera *below*ground level, the bottom depth lines and horizontal lines will merge up (become congruent) for a *very dramatic* effect.
- Angle the camera looking straight back, perpendicular to the true face. Vertical lines should be parallel. Rotate the camera on the Z axis *slightly* toward the object until the horizontal edges are also parallel. Technically, you are correcting for parallax (just a casual line to drop on your girlfriend to impress her). The example has the camera rotated 0.5 degrees toward the object.
- Move the camera toward/away from the object until it appears at the proper size relative to your passpartout.
- Adjust the Shift: Y settings until the bottom of the passpartout (or title line if you want to show some approach ground in front of

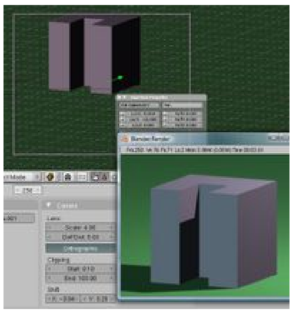
the building) is even with the bottom of the building. Adjust the X setting until the building is centered (or slightly offset from center for artistic appeal, or to show the parking lot next to it) as shown.

In the example screenshot, the Lens is 35, X is negative and Y is positive. The camera is off to the right of the object, even with the bottom of the building. If X & Y were zero, the building would have appeared off camera, in the upper left-hand corner of the passpartout.

Parallel Horizontal Edges

You can use the lines of the passpartout as a guide in rotating the camera to determine when the horizontal edges are parallel.

Orthographic Rendering


Orthographic Render

Zero point rendering is where vertical, horizontal AND depth lines are all parallel, and is commonly rendered at 45 degree, 30 degree, or 60 degree angles. With all of those sets of edges parallel to each other within that set, there are no vanishing points.

The example shows that same building rendered at 45 degrees from all angles. Note that the vertical lines are parallel to each other, the horizontals, and the depth lines are parallel to each other. From this, it is very easy to see that the left top edge of the building is the same length as the right top edge, and that the building is as deep as it is wide and high; if you measured the edges with a ruler, they would all be the same.

Orthographic rendering gives a true mathematical render of the shape of the object. An Orthographic perspective is what you see in the User View of a 3D window (if View->Orthographic is turned on).

To get an Orthographic render:

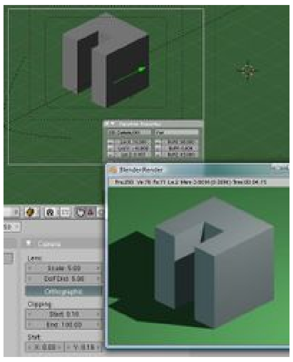
- Enable Orthographic in the Camera panel. This makes at least one face to be true to the camera.
- Point the camera at the object
- Position the camera or alter the Scale so the object is the desired size

With Orthographic cameras though, Lens size is irrelevant, since light rays do not converge to the camera from a field of view. They come in parallel, and so you can only Scale the camera size to take in more or less of that huge plane.

Note that Shift X & Y are zero, and that the camera is positioned perfectly off at a 45 degree angle to the object/building, and is rotated exactly 45 degrees to face the building. Thus, the near edge is aligned with the back edge (since the object is square).

Orthographic renders are usually made at 30, 45, or 60 degree angles to the object. Specific measurements are left to reader using triangle math.

Isometric Rendering


Isographic Render

While we are at it, we might as well cover Isometric rendering, which is a very specific type of orthographic render very often used in drafting and third-person computer games.

In Isometric renders, you want your depth lines and your horizontal lines to be at 30 degrees off horizontal, and your vertical lines to be, well, vertical. Some complicated vector calculus in Wikipedia gives us a convenient shortcut. To get Isometric Renders:

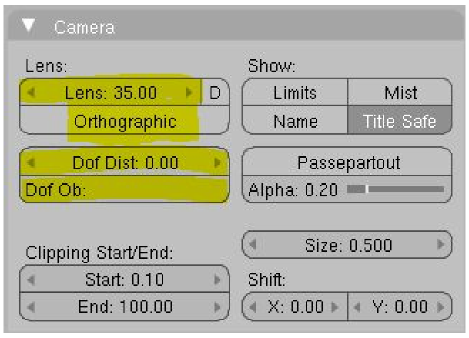
- Make your camera Orthographic

- Add a "Track To" constraint (Object F7 context, Constraints panel) to the camera for it to Track To the object (type the name in the Target OB: field), using To: -Z and Up Y.
- Position your camera so that it is 45 degrees in the XY plane from your object, and raised at a 30 degree angle. If your object is at XYZ (0,0,0), then your camera should be at (10, -10, 10), or for a view from the left side, (-10, -10, 10)
- Adjust the Scale of the camera (Editing F9 context, Camera panel) so that the object fits within the passpartout
- Adjust the Shift: Y value so that the object is centered in the render.

Depth Of Field (DOF) Explained

Real world camera lenses and your eyeball transmit light through a lens (cornea) that bends the light, and an iris that limits the amount of light, to focus the image onto the film, CCD/Cmos sensor, or retina. Because of the interaction of the lens and iris, objects that are a certain distance away are in focus; objects in the foreground and background are out of focus. We call this distance their depth, or “Z” distance from the camera or eye.

Light comes to the lens (in the real world) at an angle; from some direction. What you see depends on your perspective; if you move closer, different angles of the scene are revealed. To make “flat” pictures, like an architectural drawing or plot, Blender can also make an orthographic rendering. So, there are two kinds of renderings, Perspective and Orthographic. Perspective simulates light coming in at an angle to the lens from the field of view, and Orthographic (disabled by default) simulates light coming straight in to an infinitely large backplane or flat retina.



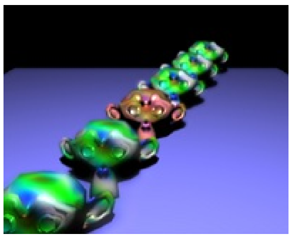
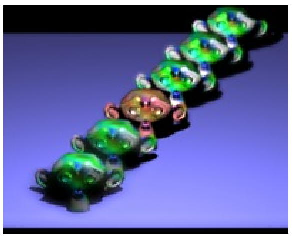
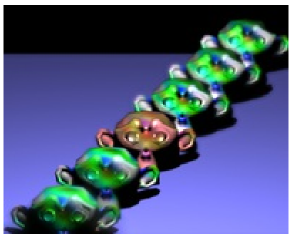
Depending on the diameter of the iris, there is a range (of distance) where objects are in focus. In cameras, the diameter of the iris is controlled by an “f-stop”. Said another way, there is *field* of view that you see left to right, up and down; your “picture”, if you will. At a certain range, or *depth* away from your eye, things are in focus. For example, at night, you may be able to focus your eye on objects that are 10 to 15 feet (3 to 5 meters) away. Anything closer than 10 or farther away than 15 is blurry. Your **depth of field** is thus 5 feet (2 m).

The larger the iris, the smaller the depth of field. This is why, during the day, you can focus on a range of things stretching out far from you. In film, there is a person whose job is to measure the distance from the camera to the actor’s nose, to ensure that the focus is set perfectly.

The more that an object is out of its depth (the perfect value for this depth is called *focal plane*), the blurrier it is. In fact, the depth of field is the range on both sides of the focal plane in which the blurriness of the objects is considered to be low enough to be imperceptible. In Blender, this distance is called the Dof Dist or “Depth of Field Distance” and is set in the Editing context (F9) for the camera. Alternatively, you can have the camera automatically stay focused on an object, by entering the name of the object in the Dof Ob field.

Field of View and Lens Size

The field of view varies by the size of the lens. With cameras, a 35mm lens is kind of a standard size because the picture it takes mimics the size of the picture seen by the eye and pictures can be taken rather close. In Blender, use the Camera settings to change the size of the lens (35mm is the default). A longer lens taking a picture farther away has the same field of view, but has a different perspective of the view that many directors love because it “condenses” the scene and smooths a sweep, since it is farther away from the action:

 35mm lens from 10 units away.	 210mm lens from 60 units away at same location/rotation.	 210mm at 50 units; repositioned to frame the view similar to the 35mm shot.

Zooming in Blender

Zoom is the ability to expand a subset of the picture; we humans have no such ability. Well, I take that back; we do: we just get up off the couch and walk up closer to what we want to see (however, this is more like “traveling” than “zooming”). Blender allows you both actions: you can move the camera closer to or farther away from an object for a track (or “truck”) in/out, and/or change its lens size. You can automate these by assigning an Interpolated (Ipo) curve to the object or to the camera, respectively.

Depth of Field in Computer Graphics

In computer graphics (CG), there is no physical lens or iris, so the depth-of-field (DOF) is infinite and all objects are always in focus. However, for artistic reasons, we want our main characters to be in focus, and everything else a little blurry, so that our audience does not focus on distracting things in the background. Also, it is easier to discern the main actors when they are in focus, and everything else isn't. So, we have to create an effect, or **Depth of Field Effect**, to composite our images and post-process them to achieve realistic-looking results.

Rendering and Saving Images

After you have adjusted your render settings, in regards to Quality and Format, you will need to actually render the image. Rendering still images is fairly simple. Rendering Animations is a bit more complex and is covered in the next sections.

To render an image from the active camera, in the Render Panel, hit the big Image button. By default the 3D view is replaced with the UV/Image Editor and the render appears.

Displaying Renders

Renders are displayed in the Image Editor. You can set the way this is displayed to several different options in the Display drop-down menu:

Keep UI
: The image is rendered to the Image Editor, but the UI remains the same. You will need to open the Image Editor manually to see the render result.

New Window
: A new floating window opens up, displaying the render.

Image Editor
: The 3D view is replaced with the Image Editor, showing the render.

Full Screen
: The Image editor replaces the UI, showing the render.

For each of these options, pressing Esc will close the render view and return to the previous view.

Saving

Rendered images can be saved by clicking on the Image menu and using the save options.

Animation Playback

The 'Play' button in the render panel will play back your rendered animation in a new window.

You can also drop images or movie files in a running animation player. It will then restart the player with the new data.

Key Short-Cuts

- A toggle frame skipping.
- P toggle ping-pong.
- ↵ Enter start playback (when paused).
- 0 NumPad toggle looping.
- Padperiod manual frame stepping.
- ← step back one frame.
- → step forward one frame.
- ↓ step back 10 frames.
- ↑ step forward 10 frames.
- ⇧ Shift↓ use backward playback.
- ⇧ Shift↑ use forward playback.
- ⇧ Shift hold to show frame numbers.

- LMB scrub in time.

- CtrlPlus zoom in
- CtrlMinus zoom out
- Esc quit

- 1 NumPad 60 fps
- 2 NumPad 50 fps
- 3 NumPad 30 fps
- 4 NumPad 25 fps
- ⇧ Shift4 NumPad 24 fps
- 5 NumPad 20 fps
- 6 NumPad 15 fps
- 7 NumPad 12 fps
- 8 NumPad 10 fps
- 9 NumPad 6 fps
- / NumPad 5 fps
- Minus slow down playback.
- Plus speed up playback.

Display Options

When a rendered image is displayed in the Image Editor, several new menu items become available.

Slot Menu
: You can save successive renders into the render buffer by selecting a new slot before rendering. If an image has been rendered to a slot, it can be viewed by selecting that slot. Empty slots appear as blank grids in the image editor. Use the shortcut J to

cycle through saved renders and AltJ to cycle backwards through the saved renders.

Render Layer
: If you are using Render Layers, use this menu to select which layer is displayed.

Render Pass
: If you are using Render Passes, use this menu to select which pass is displayed.

Image Painting
: This icon enables or disables Image Painting.

Display Mode
: The last four buttons set how the image is displayed.

  RGB

    Draw image as rendered, without alpha channel.

  RGBA

    Replaces transparent pixels with background checkerboard, denoting the alpha channel.

  Alpha Channel

    Displays a gray-scale image. White areas are opaque, black areas have a an alpha of 0.

  Z Depth

    Display the depth from the camera, from Clip Start to Clip End, as specified in the Camera settings.

Curves Panel
: The Curves Panel is available in the Properties Panel. You can use this to adjust the colors of the image.

Render Quality

Many factors go into the quality of the rendered image. Rendering a scene without changing any of the render settings is probably going to produce a rather unpleasant image. In previous chapters, you have learned how to model, shade, texture, and light scenes. Optimizing settings with respect to those areas will help to produce quality images, but there are some important settings that come into play before hitting the render button. These can directly affect the look of the rendered image.

The next section covers render layers and render passes, both of which allow you to compose an image from several elements of a scene. In some cases it is necessary to render effects straight out of the renderer, rather than creating them in "post."

Color Management and Exposure

One important aspect of 3D rendering that is often overlooked is color management. Without color management, or more commonly, linear rendering, render engines interpret scene lighting correctly, but display them incorrectly on your monitor. Blender simplifies this workflow, but it is important to know how the color space of a rendered image factors into your pipeline.

Anti-Aliasing

Anti-Aliasing removes jagged edges that appear in contrasting areas of color. This is a very important aspect of render quality. Without this render setting, images usually appear particularly CG and amateur.

Exposure (Lighting)

Exposure is, in physical terms, the length of time a camera's film or sensor is exposed to light. Longer exposure times create a brighter image. In CG, the recorded light values are offset to simulate longer or shorter exposures. This can be achieved through lighting settings, or better, through Color Management settings

Depth of Field

Real cameras have a specific focal length. This is the distance from the lens where everything is in focus. Certain factors determine how much objects out of this range, or depth of field, are out of focus. By default, when rendering, all objects appear in perfect focus. Depth of Field (DOF) can create an unusual or appropriate sense of scale, depending how it is used.

Motion Blur

Cameras have a certain shutter speed, or the length of time the film is exposed to the image. Things that are in motion while the picture is taken will have some degree of blurring. Faster-moving objects will appear more blurred than slower objects. This is an important effect in CG because it is an artifact that we expect to see, and when it is missing, an image may not be believable.

Anti-Aliasing

A computer generated image is made up of pixels; each pixel can of course only be a single color. In the rendering process the rendering engine must therefore assign a single color to each pixel on the basis of what object is shown in that pixel. This often leads to poor results, especially at sharp boundaries, or where thin lines are present, and it is particularly evident for oblique lines.

To overcome this problem, which is known as *Aliasing*, it is possible to resort to an Anti-Aliasing technique. Basically, each pixel is 'oversampled', by rendering it as if it were 5 pixels or more, and assigning an 'average' color to the rendered pixel.

The buttons to control Anti-Aliasing, or OverSampling (OSA), are below the rendering button in the Render Panel (*Render Panel.*).

Options

Anti-Aliasing check box
    Enables oversampling

5 / 8 / 11 / 16
    The number of samples to use. The values 5, 8, 11, 16 are preset numbers in specific sample patterns; a higher value produces better edges, but slows down the rendering.

By default, we use in Blender a fixed "Distributed Jitter" table. The samples within a pixel are distributed and jittered in a way that guarantees two characteristics:

1. Each sample has equal distances to its neighbor samples
2. The samples cover all sub-pixel positions equally, both horizontally and vertically

   The images below show Blender sample patterns for 5, 8, 11 and 16 samples. To show that the distribution is equalized over multiple pixels, the neighbor pixel patterns were drawn as well. Note that each pixel has an identical pattern.

5 samples   8 samples   11 samples   16 samples

Full Sample

    For every anti-aliasing sample, save the entire Render Layer results. This solves anti-aliasing issues with compositing.

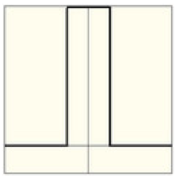
Filtering

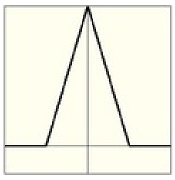
When the samples have been rendered, we've got color and alpha information available per sample. It then is important to define how much each sample contributes to a pixel.

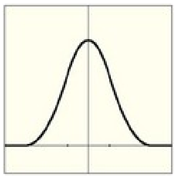
The simplest method is to average all samples and make that the pixel color. This is called using a "Box Filter". The disadvantage of this method is that it doesn't take into account that some samples are very close to the edge of a pixel, and therefore could influence the color of the neighbor pixel(s) as well.

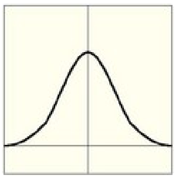
Filter menu: Set The filter type to use to 'average' the samples: |Box |The original filter used in Blender, relatively low quality. For the Box Filter, you can see that only the samples within the pixel itself are added to the pixel's color. For the other filters, the formula ensures that a certain amount of the sample color gets distributed over the other pixels as well.

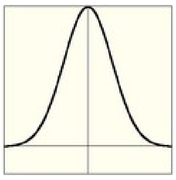
Box	A low-quality box-shaped curve (see above)
Tent	A simplistic filter that gives sharp results
Quadratic	A Quadratic curve
Cubic	A Cubic curve
Gauss	Gaussian distribution, the most blurry
Catmull-Rom	Catmull-Rom filter, gives the most sharpening
Mitchell-Netravali	Mitchell-Netravali, a good all-around filter that gives reasonable sharpness

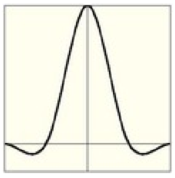

Box

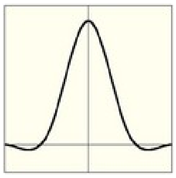

Tent


Quadratic


Cubic





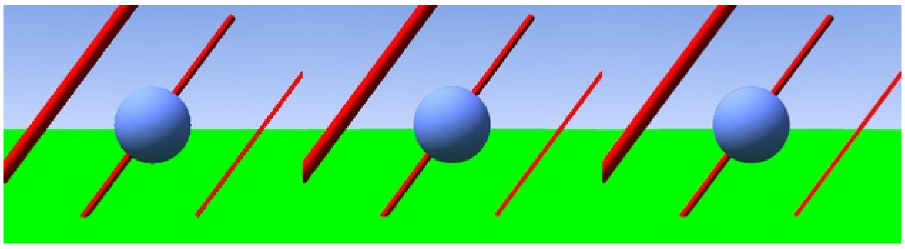


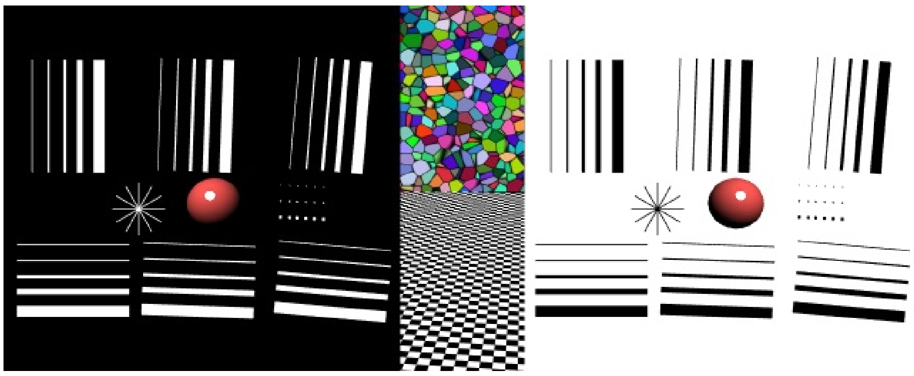
Gaussian Catmull-Rom Mitchell-Netravali

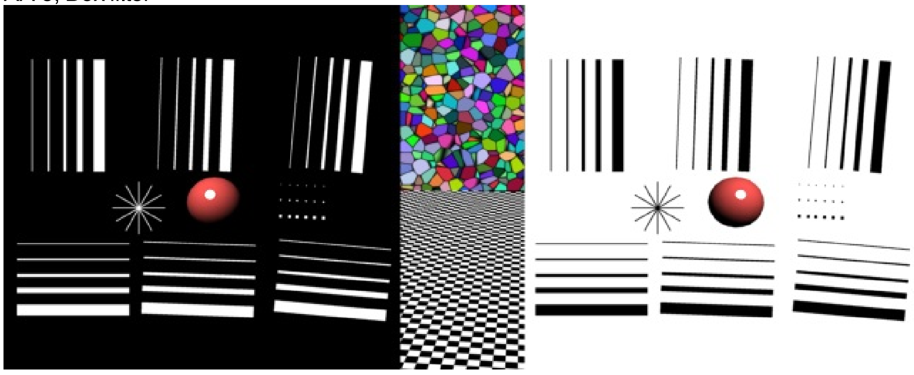
Filter Size

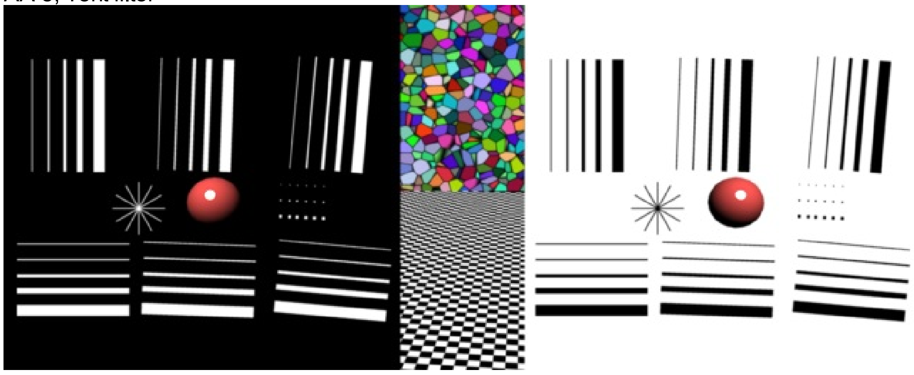
Making the filter size value smaller will squeeze the samples more into the center, and blur the image more. A larger filter size makes the result sharper. Notice that the last two filters also have a negative part; this will give an extra sharpening result.

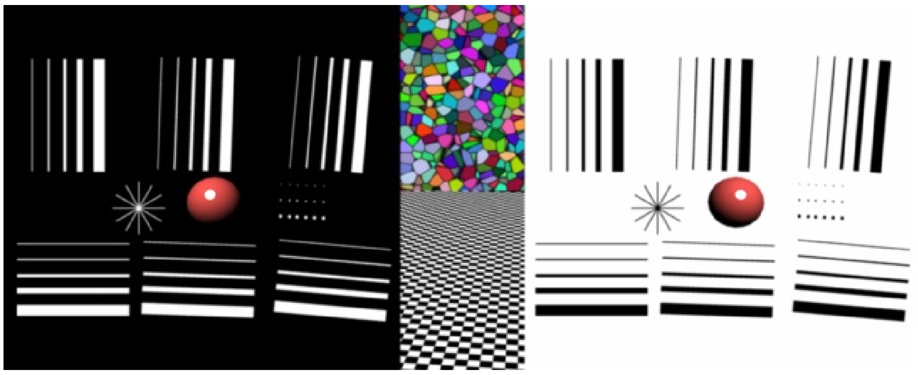
Examples

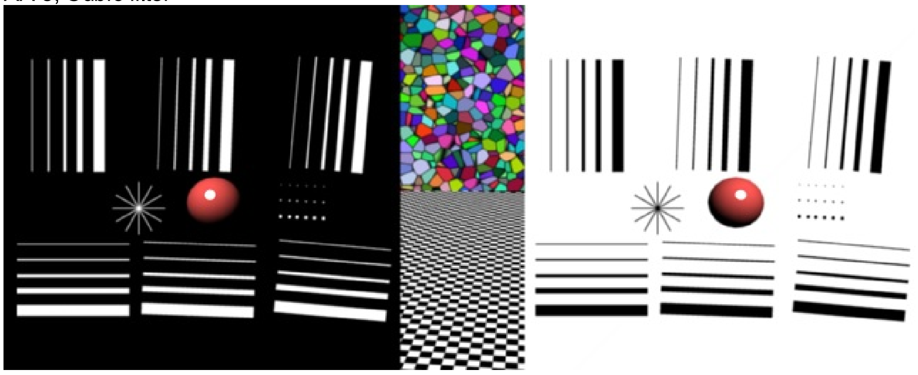

Rendering without AA (left) with AA=5 (center) and AA=8 (right).

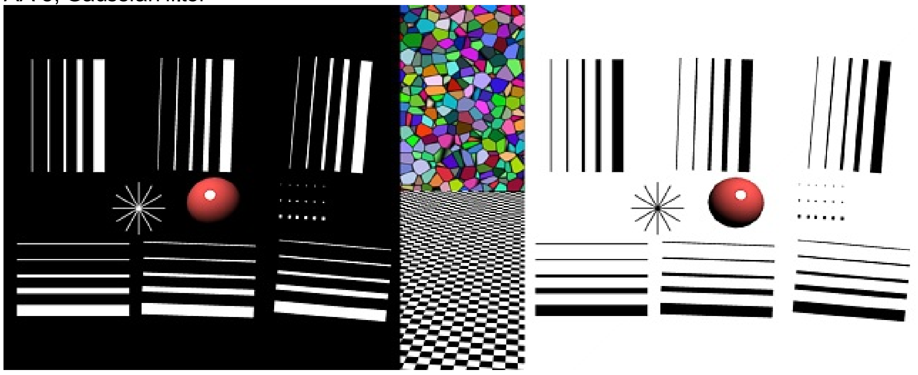

AA 8, Box filter

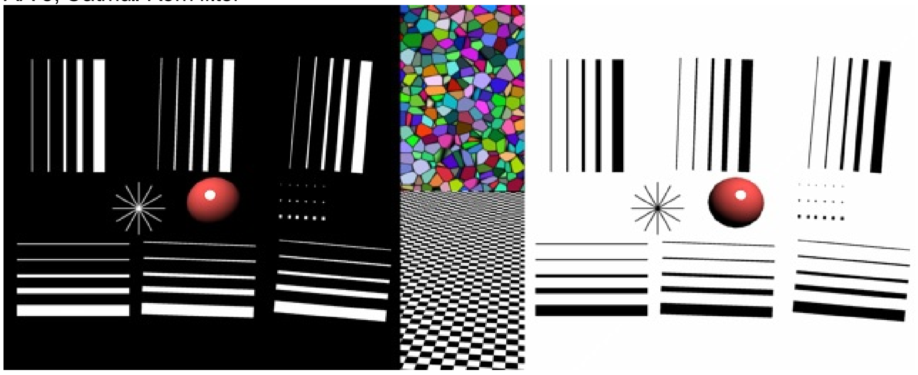

AA 8, Tent filter


AA 8, Quadratic filter


AA 8, Cubic filter


AA 8, Gaussian filter


AA 8, Catmull-Rom filter


AA 8, Mitchell-Netravali filter

Rendering Animations

While rendering stills will allow you to view and save the image from the render buffer when it's complete, animations are a series of images, or frames, and are automatically saved directly out to disk after being rendered.

After rendering the frames, you may need to edit the clips, or first use the Compositor to do green-screen masking, matting, color correction, DOF, and so on to the images. That result is then fed to the Sequencer where the strips are cut and mixed and a final overlay is done.

Finally you can render out from the Sequencer and compress the frames into a playable movie clip.

Workflow

Generally, you do a lot of intermediate renders of different frames in your animation to check for timing, lighting, placement, materials, and so on. At some point, you are ready to make a final render of the complete animation for publication.

There are two approaches you can use when making a movie, or animation, with or without sound. The approach you should use depends on the amount of CPU time you will need to render the movie (see Render Performance). You can render a "typical" frame at the desired resolution, and then multiply by the number of frames that will ultimately go into the movie, to arrive at an total render time.

If the total render time is an hour or more, you want to use the "Frame Sequence" approach. For example, if you are rendering a one-minute video clip for film, there will be (60 seconds per minute) * (24 frames per second) or 1440 frames per minute. If each frame takes 30 seconds to render, then you will be able to render two frames per minute, or need 720 minutes (12 hours) of render time.

Rendering takes all available CPU time; you should render overnight, when the computer is not needed, or set Blender to a low priority while rendering, and work on other things (be careful with the RAM space!).

The **Direct Approach**—highly **not** recommended and not a standard practice—is where you set your output format to an AVI or MOV format, and click ANIM to render your scene directly out to a movie file. Blender creates one file that holds all the frames of your animation. You can then use Blender's VSE to add an audio track to the animation and render out to an MPEG format to complete your movie.

The **Frame Sequence** is a much more stable approach, where you set your output format to a still format (such as JPG, PNG or MultiLayer), and click ANIM to render your scene out to a set of images, where each image is the frame in the sequence.

Blender creates a file for each frame of the animation. You can then use Blender's compositor to perform any frame manipulation (post processing). You can then use Blender's VSE to load that final image sequence, add an audio track to the animation, and render out to an MPEG format to complete your movie. The Frame Sequence approach is a little more complicated and takes more disk space, but gives you more flexibility.

Here are some guidelines to help you choose an approach.

Direct Approach

- short segments with total render time < 1 hour
- stable power supply
- computer not needed for other uses

Frame Sequence Approach

- total render time > 1 hour
- post-production work needed
    - Color/lighting adjustment
    - Green screen / matte replacement
    - Layering/compositing
    - Multiple formats and sizes of ultimate product
- intermediate frames/adjustments needed for compression/codec
- precise timing (e.g. lip-sync to audio track) needed in parts
- may need to interrupt rendering to use the computer, and want to be able to resume rendering where you left off.

Frame Sequence Workflow

1. First prepare your animation.
2. In the Dimensions panel, choose the render size, Pixel Aspect Ratio, and the Range of Frames to use, as well as the frame rate, which should already be set.
3. In the Output panel set up your animation to be rendered out as as images, generally using a format that does not compromise any quality (I prefer PNG or MultiLayer because of their loss-less nature).
4. Choose the output path and file type in the Output panel as well, for example "//\render\my-anim-".
5. Confirm the range of your animation frame Start and End.
6. Save your .blend file.
7. Press the big *Animation* button. Do a long task [like sleeping, playing a video game, or cleaning your driveway] while you wait for your computer to finish rendering the frames.
8. Once the animation is finished, use your OS file explorer to navigate into the output folder (".\render in this example). You will see lots of images (.png or .exr, etc... depending on the format you chose to render) that have a sequence number attached to them ranging from 0000 to a max of 9999. These are your single frames.
9. In Blender, now go into the video sequence editor.
10. Choose *Add Image* from the add menu. Select all the frames from your output folder that you want to include in your animation

(Press A to Select All easily). They will be added as a strip to the sequence editor.

11. Now you can edit the strip and add effects or simply leave it like it is. You can add other strips, like an audio strip.
12. Scrub through the animation, checking that you have included all the frames.
13. In the Scene Render buttons, in the Post Processing panel, activate *Sequencer*.
14. In the Format panel, choose the container and codec you want (e.g. MPEG H.264) and configure it. The video codecs are described on the previous page: Output Options.
15. Click the ANIMATION render button and Blender will render out the sequence editor output into your movie.

Why go through all this hassle? Well, first of all, if you render out single frames you can stop the render at any time by pressing Esc in the render window. You will not lose the frames you have already rendered, since they have been written out to individual files. You can always adjust the range you want to continue from where you left off.

You can edit the frames afterwards and post-process them. You can add neat effects in the sequence editor. You can render the same sequence into different resolutions (640x480, 320x240, etc) and use different codecs (to get different file sizes and quality) with almost no effort whatsoever.

Options

Output Panel

By default the animation is rendered in the directory specified in the Output Panel (*Animation location and extensions.*). If an AVI format has been selected, then the name will be ####_####.avi where the '####' indicates the start and end frame of the animation, as 4 digit integers padded with zeros as necessary.

If an image format is chosen, on the other hand, a series of images named ####, ('####' being the pertinent frame number) is created in the directory.

File Extensions
: Adds the correct file extensions per file type to the output files

Overwrite
: Overwrite existing files when rendering

Placeholders
: Create empty placeholder frames while rendering

Post Processing Panel

Sequencer
: Renders the output of the sequence editor, instead of the view from the 3D scene's active camera. If the sequence contains scene strips, these will also be rendered as part of the pipeline. If Do Composite is also enabled, the Scene strip will be the output of the Compositor.

Compositing
: Renders the output from the Compositing noodle, and then pumps all images through the Composite node map, displaying the image fed to the Composite Output node.

Hints

Argh! My bratty sister turned off the PC right in the middle of rendering my movie!
: Unless your animation is really simple, and you expect it to render in half an hour or less, it is always a good idea to render the animation as separate image frames in a loss-less format (TGA, PNG, BMP) rather than as a movie file from the beginning. This allows you an easy recovery if there is a problem and you have to re-start the rendering, since the frames you have already rendered will still be in the Output directory. Just change the START frame number to the frame number where you want to pick up from, and click ANIM again.

I only need to re-render a few frames in the middle
: It's also a good idea to render initially to a frame sequence, since if only a few frames have an error, you can make corrections and re-render just the affected frames. You can then make a movie out of the separate frames with Blender's sequence editor or with compositing nodes.

Only first frame renders, then Blender locks up
: If you click ANIM and only the first frame renders, be sure the output file is not locked by the media player. In general, check the console when rendering.

Unable to create Quicktime movie
: CreateMovieFile error: -47
: The Quicktime movie strip is in use (possibly in the VSE) and cannot be overwritten. If it is used in the VSE, delete the strip, or delete the file using your file explorer.

Render Baking

Baking, in general, is the act of pre-computing something in order to speed up some other process later down the line. Rendering from scratch takes a lot of time depending on the options you choose. Therefore, Blender allows you to "bake" some parts of the render ahead of time, for select objects. Then, when you press Render, the entire scene is rendered much faster, since the colors of those objects do not have to be recomputed.

Render baking creates 2D bitmap images of a mesh object's rendered surface. These images can be re-mapped onto the object using the object's UV coordinates. Baking is done for each individual mesh, and can only be done if that mesh has been UV-unwrapped. While it takes time to set up and perform, it saves render time. If you are rendering a long animation, the time spent baking can be much less than time spent rendering out each frame of a long animation.

Use Render Bake in intensive light/shadow solutions, such as AO or soft shadows from area lights. If you bake AO for the main objects, you will not have to enable it for the full render, saving render time.

Use Full Render or Textures to create an image texture; baked procedural textures can be used as a starting point for further texture painting. Use Normals to make a low-resolution mesh look like a high-resolution mesh. To do that, UV-unwrap a high-resolution, finely sculpted mesh and bake its normals. Save that normal map, and Mapping (texture settings) the UV of a similarly unwrapped low-resolution mesh. The low-resolution mesh will look just like the high-resolution, but will have much fewer faces/polygons.

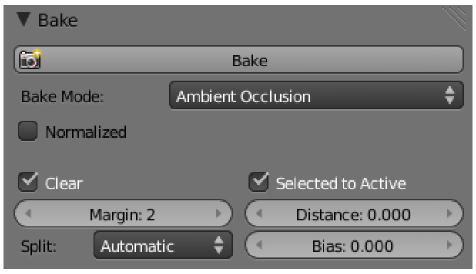
Advantages

- Can significantly reduce render times
- Texture painting made easier
- Reduced polygon count
- Repeated renders are made faster, multiplying the time savings

Disadvantages

- Object must be UV-unwrapped.
- If shadows are baked, lights and object cannot move with respect to each other.
- Large textures (eg 4096x4096) can be memory intensive, and be just as slow as the rendered solution.
- Human (labor) time must be spent unwrapping and baking and saving files and applying the textures to a channel.

Options



Ambient Occlusion

Bake Mode

Full Render

Bakes all materials, textures, and lighting except specularity and SSS.

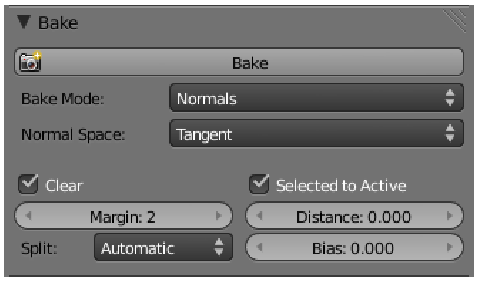
Ambient Occlusion

Bakes ambient occlusion as specified in the World panels. Ignores all lights in the scene.

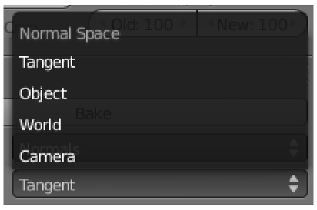
Normalized
: Normalize without using material's settings.

Shadow

Bakes shadows and lighting.



Normals



Normal Space

Normals

Bakes tangent and camera-space normals (amongst many others) to an RGB image.

Normal Space
: Normals can be baked in different spaces:

  Camera space
  : Default method.

  World space
  : Normals in world coordinates, dependent on object transformation and deformation.

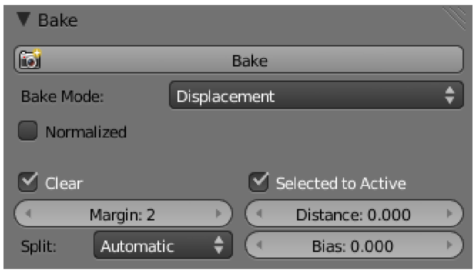
  Object space
  : Normals in object coordinates, independent of object transformation, but dependent on deformation.

  Tangent space
  : Normals in tangent space coordinates, independent of object transformation and deformation. This is the new default, and the right choice in most cases, since then the normal map can be used for animated objects too.

For materials the same spaces can be chosen as well, in the image texture options, next to the existing Normal Map setting. For correct results, the setting here should match the setting used for baking.

Textures

Bakes colors of materials and textures only, without shading.


Bake
Bake
Bake Mode: Displacement
Normalized
Clear ✓
Selected to Active ✓
Margin: 2
Distance: 0.000
Split: Automatic
Bias: 0.000

Displacement

Displacement

Similar to baking normal maps, displacement maps can also be baked from a high-res object to an unwrapped low-res object, using the Selected to Active option.

Normalized
: Normalize to the distance.

When using this in conjunction with a subsurf and displacement modifier within Blender, it's necessary to temporarily add a heavy subsurf modifier to the 'low res' model before baking. This means that if you then use a displacement modifier on top of the subsurf, the displacement will be correct, since it's stored as a relative difference to the subsurfed geometry, rather than the original base mesh (which can get distorted significantly by a subsurf). The higher the render level subsurf while baking, the more accurate the displacements will be. This technique may also be useful when saving the displacement map out for use in external renderers.

Emission

Bakes Emit, or the Glow color of a material.

Alpha

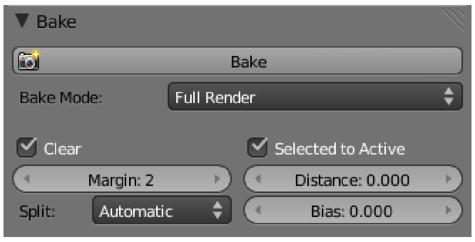
Bakes Alpha values, or transparency of a material.

Mirror Color and Intensity

Bakes Mirror color or intensity values.

Specular Color and Intensity

Bakes specular color or specular intensity values.



Full Render

Additional Options

Clear
    If selected, clears the image to selected background color (default is black) before baking render.
Margin
    Baked result is extended this many pixels beyond the border of each UV "island," to soften seams in the texture.

Split
    Fixed

        Slit quads predictably (0,1,2) (0,2,3).

    Fixed alternate

        Slit quads predictably (1,2,3) (1,3,0).

    Automatic

        Split quads to give the least distortion while baking.

Select to Active
    Enable information from other objects to be baked onto the active object.

    Distance
        Controls how far a point on another object can be away from the point on the active object. Only needed for Selected to Active.
        A typical use case is to make a detailed, high poly object, and then bake it's normals onto an object with a low polygon count. The resulting normal map can then be applied to make the low poly object look more detailed.
    Bias
        Bias towards further away from the object (in blender units)

Mesh Must be Visible in Render
If a mesh is not visible in regular render, for example because it is disabled for rendering in the Outliner or has the DupliVerts setting enabled, it cannot be baked to.

Workflow

1. In a 3D View window, select a mesh and enter UV/Face Select mode
2. Unwrap the mesh object
3. In a UV/Image Editor window, either create a new image or open an existing one. If your 3D view is in textured display mode, you should now see the image mapped to your mesh. Ensure that all faces are selected.
4. In the Bake panel at the bottom of the Render menu, bake your desired type of image (Full Render etcetera.)
5. When rendering is complete, Blender replaces the image with the Baked image.
6. Save the image.
7. Apply the image to the mesh as a UV texture. For displacement and normal maps, refer to Bump and Normal Maps. For full and texture bakes, refer to Textures.
8. Refine the image using the process described below, or embellish with Texture Paint or an external image editor.

Introduction

In some situations we want to increase the render speed, access blender remotely to render something or build scripts that use blender command line.

One advantage of using command line is that we don't need the X server (in case of Linux) and as a consequence we can render remotely by SSH or telnet.

Note! Arguments are executed in the order they are given!

```
blender -b file.blend -a -x 1 -o //render
```

...Wont work, since the output and extension is set after blender is told to render.

Always position **-f** or **-a** as the last arguments.

Syntax

```
blender [-b <dir><file> [-o <dir><file>][-F <format>]
[-x [0|1]][-t <threads>][-S <name>][-f <frame>]
[-s <frame> -e <frame> -a]] [[-P <scriptname> [-- <parameter>]]
```

Render Options:

```
-b or --background <file>
    Load <file> in background (often used for UI-less rendering)

-a or --render-anim
    Render frames from start to end (inclusive)

-S or --scene <name>
    Set the active scene <name> for rendering

-f or --render-frame <frame>
    Render frame <frame> and save it.
    +<frame> start frame relative, -<frame> end frame relative.

-s or --frame-start <frame>
    Set start to frame <frame> (use before the -a argument)

-e or --frame-end <frame>
    Set end to frame <frame> (use before the -a argument)

-j or --frame-jump <frames>
    Set number of frames to step forward after each rendered frame

-o or --render-output <path>
    Set the render path and file name.
    Use // at the start of the path to
        render relative to the blend file.
    The # characters are replaced by the frame number, and used to define zero padding.
        ani_##_test.png becomes ani_01_test.png
        test-######.png becomes test-000001.png
        When the filename does not contain #, The suffix #### is added to the filename
    The frame number will be added at the end of the filename.
        eg: blender -b foobar.blend -o //render_ -F PNG -x 1 -a
        //render_ becomes //render_####, writing frames as //render_0001.png//

-E or --engine <engine>
    Specify the render engine
    use -E help to list available engines

-t or --threads <threads>
    Use amount of <threads> for rendering and other operations
    [1-64], 0 for systems processor count.
```

Format Options:

```
-F or --render-format <format>
    Set the render format, Valid options are...
        TGA IRIS JPEG MOVIE IRIZ RAWTGA
        AVIRAW AVIJPEG PNG BMP FRAMESERVER
    (formats that can be compiled into blender, not available on all systems)
        HDR TIFF EXR MULTILAYER MPEG AVICODEC QUICKTIME CINEON DPX DDS

-x or --use-extension <bool>
    Set option to add the file extension to the end of the file
```

Animation Playback Options:

```
-a <options> <file(s)>
    Playback <file(s)>, only operates this way when not running in background.
```

```
        -p <sx> <sy>    Open with lower left corner at <sx>, <sy>
        -m      Read from disk (Don't buffer)
        -f <fps> <fps-base>     Specify FPS to start with
        -j <frame>  Set frame step to <frame>
        -s <frame>  Play from <frame>
        -e <frame>  Play until <frame>
```

Window Options:

```
-w or --window-border
    Force opening with borders (default)

-W or --window-borderless
    Force opening without borders

-p or --window-geometry <sx> <sy> <w> <h>
    Open with lower left corner at <sx>, <sy> and width and height as <w>, <h>

-con or --start-console
    Start with the console window open (ignored if -b is set), (Windows only)

--no-native-pixels
    Do not use native pixel size, for high resolution displays (MacBook 'Retina')
```

Game Engine Specific Options:

```
-g Game Engine specific options
    -g fixedtime        Run on 50 hertz without dropping frames
    -g vertexarrays     Use Vertex Arrays for rendering (usually faster)
    -g nomipmap     No Texture Mipmapping
    -g linearmipmap     Linear Texture Mipmapping instead of Nearest (default)
```

Python Options:

```
-y or --enable-autoexec
    Enable automatic python script execution

-Y or --disable-autoexec
    Disable automatic python script execution (pydrivers & startup scripts), (compiled as non-standard default)


-P or --python <filename>
    Run the given Python script file

--python-text <name>
    Run the given Python script text block

--python-console
    Run blender with an interactive console

--addons
    Comma separated list of addons (no spaces)
```

Debug Options:

```
-d or --debug
    Turn debugging on

    * Prints every operator call and their arguments
    * Disables mouse grab (to interact with a debugger in some cases)
    * Keeps python sys.stdin rather than setting it to None

--debug-value <value>
    Set debug value of <value> on startup



--debug-events
    Enable debug messages for the event system

--debug-handlers
    Enable debug messages for event handling

--debug-jobs
    Enable time profiling for background jobs.

--debug-python
    Enable debug messages for python

--debug-wm
    Enable debug messages for the window manager

--debug-all
```

```
    Enable all debug messages (excludes libmv)


--debug-fpe
    Enable floating point exceptions

--disable-crash-handler
    Disable the crash handler
```

Misc Options:

```
--factory-startup
    Skip reading the "startup.blend" in the users home directory


--env-system-datafiles
    Set the BLENDER_SYSTEM_DATAFILES environment variable

--env-system-scripts
    Set the BLENDER_SYSTEM_SCRIPTS environment variable

--env-system-python
    Set the BLENDER_SYSTEM_PYTHON environment variable


-nojoystick
    Disable joystick support

-noglsl
    Disable GLSL shading

-noaudio
    Force sound system to None

-setaudio
    Force sound system to a specific device
    NULL SDL OPENAL JACK


-h or --help
    Print this help text and exit

-v or --version
    Print Blender version and exit

--
    Ends option processing, following arguments passed unchanged. Access via python's sys.argv
```

Other Options:

```
/?
    Print this help text and exit (windows only)

--verbose <verbose>
    Set logging verbosity level.

-R
    Register .blend extension, then exit (Windows only)

-r
    Silently register .blend extension, then exit (Windows only)
```

Examples

Render a picture

```
# blender -b file.blend -o //file -F JPEG -x 1 -f 1
```

- **-b**

  Load blender without an interface
- **file.blend**

  File .blend to render
- **-o //file**

  Directory + Target image file
- **-F JPEG**

  JPEG image format

- **-x 1**

  Ensures an extension .jpg to the file name
- **-f 1**

  Render frame 1

Render a movie

```
# blender -b file.blend -x 1 -o //file -F MOVIE -s 003 -e 005 -a
```

- **-b**

  Load blender without an interface
- **file.blend**

  File .blend to render
- **-x**

  Ensures an extension .avi to the movie
- **-o //file**

  Directory + Target image file
- **-F MOVIE**

  This saves a .AVI movie with low compression
- **-s 003 -e 005 -a**

  Set start frame to 003 and end frame to 005. ***Important:*** *You can use -s or -e, but if they're not in order, they'll not work!*

Launch Blender with a specified engine

The flags that are used are -E engine or --engine engine

```
# blender --engine CYCLES
```

You can list the available engines by using

```
# blender --engine help
```

or

```
# blender -E help
```

Example output

```
found bundled python: /home/satishg/bin/blender-2.65a-linux-glibc27-x86_64/2.65/python
Blender Engine Listing:
       BLENDER_RENDER
       BLENDER_GAME
       CYCLES[1]    Done                          ~/bin/blender-2.65a-linux-glibc27-x86_64/blender --engine help
```

Platforms

How to actually execute Blender from the command line depends on the platform and where you have installed Blender. Here are basic instructions for the different platforms.

Windows

Open the Command Prompt, go to the directory where Blender is installed, and then run the blender command.

```
# cd c:\<blender installation directory>
# blender
```

Mac OS X

Open the Terminal application, go to the directory where Blender is installed, and run the executable within the app bundle, with commands like this:

```
# cd /Applications/Blender
# ./blender.app/Contents/MacOS/blender
```

If you need to do this often, you can make an alias so that typing just 'blender' in the terminal works. For that you can run a command like this in the terminal (with the appropriate path).

```
# echo "alias blender=/Applications/Blender/blender.app/Contents/MacOS/blender" >> ~/.profile
```

If you then open a new terminal, the following command will work:

```
# blender
```

Linux

Open a terminal, then go to the directory where Blender is installed, and run the blender command like this.

```
# cd <blender installation directory>
# ./blender
```

If you have Blender installed in your PATH (usually when Blender is installed through a linux distribution package), you may be able to simple do this:

```
# blender
```

Output Options

The first step in the rendering process is to determine and set the output options. This includes render size, frame rate, pixel aspect ratio, output location, and file type.

Dimensions

Resolution
: The Dimensions section has settings for the size of the rendered images.
By default the dimensions SizeX and SizeY are 1920×1080 and can be changed by adjusting the X and Y fields. These buttons control the overall size of the image.

The Percentage slider will scale the currently set resolution by that value. This is useful for small test renders that are the same proportions as the final image.

Aspect Ratio
: Just below are two more settings, AspX and AspY which control the shape of the pixels along the respective axis. By default it is 1:1 since computer screen pixels are square. If television shorts are being made, and since TV pixels are not square, you want to change this aspect ratio to match the destination video standard: PAL for Europe, and NTSC for the Americas.

See Video Output for details on pixel aspect ratio.

Border
: You can render just a portion of the view instead of rendering the entire frame. While in Camera View, enable Border and press CtrlB, then drag a rectangle to define the area you want to render. CtrlAltB is the shortcut to disable the border.
Note that this disables the Save Buffers option in Performance and Full Sample option in Anti-Aliasing.

Enabling Crop will crop the rendered image to the Border size, instead of rendering a black region around it.

Frame Range
: Set the Start and End frames for Rendering Animations. Step controls the number of frames to advance by for each frame in the timeline.

Frame Rate
: For an Animation the frame rate, or how many frames will be displayed per second, which, by default, is 24 frames per second, the standard for animation. Use 29.97 frames per second for USA television.

Time Remapping
: Use to remap the length of an animation.

Presets

To make life easier the topmost menu provides some common presets (par = Pixel Aspect Ratio). You can add your own or remove one with the + and - buttons:

DVCPRO HD 1080p	1280x1080, 3:2par 24fps
DVCPRO HD 720p	960x720 4:3par 24fps
HDTV 1080p	1920×1080 square pixels 24fps
HDTV 720p	1280x720 square pixels 24fps
HDV 1080p	1440x1080 4:3par 23.98fps
HDV NTSC 1080p	1440x1080 4:3par 29.97fps
HDV PAL 1080p	1440x1080 4:3par 25fps
TV NTSC 16:9	720x480 4:3.3par 29.97fps
NTSC 4:3	720×480 10:11par. 29.97fps
PAL 16:9	720x576 16:11par 25fps
PAL 4:3	720x576 12:11par 25fps

These are just the presets; you can set any resolution you wish, subject to your PC's memory restrictions; see the Render page for ideas and techniques and tools for enabling huge render outputs.

Output Panel

This panel provides options for setting the location of rendered frames for animations, and the quality of the saved images.

File Locations

By default, each frame of an animation is saved in the /tmp directory. Change this or any field by ⇧ Shift LMB clicking in the name field and entering a new name. If you use the // and do not save a new .blend file somewhere, Blender assumes the // to refer to the Blender install folder.

Clicking the folder icon to the right of the field turns a Blender window pane into a File Browser window. This window is very handy for scrolling through your hard disk and selecting a file or directory.

PathSpecs
The path specification for the location can be absolute *On Microsoft-Windows include a normal or mapped drive letter (e.g. "F:")*, a breadcrumb notation (e.g. "./" and "../" and "//" (the blend file location). Forward slashes (Unix-style) or backslashes (Windows-style) are acceptable on either platform. If omitted, the file is saved in the current working directory blender was started from.

File Type

Blender supports a wide mix of image formats. These formats are listed in alphabetical order.

The output format for Animations **Animation** CtrlF12 is selected using the File Format Menu. From here you can select many image or animation formats. When rendering static images, you can select the file type after you render when you save the image.

There are many image formats out there for many different uses. A format stores an image in a *loss-less* or lossy format; with lossy formats you suffer some image degradation but save disk space because the image is saved using fewer bytes. A loss-less format preserves the image exactly, pixel for pixel. You can break formats down into *static* images and movie *clips*.

Within either category there are standards (static formats and clip codecs) which may be proprietary standards (developed and controlled by one company), or open standards (which are community or consortium-controlled). Open standards generally outlive any one particular company and will always be royalty-free and freely obtained by the viewer. Proprietary formats may only work with a specific video card, or while the codec may be free, the viewer may cost.

Compression

Some formats can compress the image to use less disk space. This compression might be loss-less (PNG, ...) or lossy (Jpeg, ...). Lossy formats don't store individual pixel information, thus reducing image quality. All the other formats are more or less equivalent, each having advantages and disadvantages. Make your compression selection using the button or field located beneath the format selector. For example, if Jpeg is selected, you can specify a compression level (Quality:90 by default). Higher quality takes more disk space, but results in a better looking picture with less compression encoding artifacts.

The default image type is Targa, but, since the image is stored in a buffer and then saved, it is possible to change the image file type after the rendering and before saving using this menu. (**Attention**: this is only valid for static images, not when rendering animations!).

Channels

Blender renders color (RGB) images by default, but Black and White (BW) and color with Alpha Channel (RGBA) are also possible. Beware: unless the Extensions button of the Output panel is set, Blender does *not* automatically add extensions to filenames, hence any .tga or .png extension must be explicitly written in the File Save window.

OpenEXR and **OpenEXR Multilayer** formats are the only formats that store Z-depth buffer information. **OpenEXR Multilayer** is the only format that stores Render Layer and Render Passes as layers that can then be composited in post-production.

Image Formats

BMP	Bit-Mapped Paint loss-less format used by early paint programs.
Iris	The standard Silicon Graphics Inc (SGI) format used on those spanking Unix OS machines.
PNG	Portable Network Graphics, a standard meant to replace old GIF inasmuch as it is loss-less, but supports full true color images. Supports Alpha channel. Enable the RGBA button to save the Alpha channel.
Jpeg	Joint Picture Expert Group (name of the consortium which defined it), an open format that supports very good compression with little loss of quality. Only saves RGB values. Re-saving images results in more and more compression and loss of quality.
Jpeg 2000	Uses the Jpeg 2000 codec.
TARGA and Targa raw	Truevision Advanced Raster Graphics Adapter is a simple raster graphics format established in 1984 and used by the original IBM PCs. Supports Alpha Channel. Enable the RGBA button to save the Alpha channel.
Cineon	format produced by a Kodak Cineon camera and used in high-end graphics software and more directed toward digital film.
DPX	Digital Moving-Picture eXchange format; an open professional format (close to Cineon) that also contains metainformation about the picture; 16-bit uncompressed bitmap (huge file size). Used in preservation.
MultiLayer	an OpenEXR format that supports storing multiple layers of images together in one file. Each layer stores a render pass, such as shadow, specularity, color, etc. You can specify the encoding used to save the MultiLayer file using the codec selector (ZIP (loss-less) is shown and used by default).
OpenEXR	an open and non-proprietary extended and highly dynamic range (HDR) image format, saving both Alpha and Z-depth buffer information. • Enable the *Half* button to use the 16-bit format; otherwise 32-bit floating point precision color depth will be used. • Enable the *Zbuf* button to save the Z-buffer (distance from camera) info. • Choose a compression/decompression *CODEC* (ZIP by default) to save disk space. • Enable the *RGBA* button to save the Alpha channel.

- Because OpenEXR is so new and previews are generally not supported by Operating Systems, enable *Preview*to save a JPG image along with the EXR image so you can quickly and easily see what the basic image looks like.

Radiance HDR	a High Dynamic Range image format that can store images in floating point (with light brighter than 1.0) - 32bits per channel.
TIFF	Often used for teletype and facsimile (FAX) images.
Frame Server	This is an alternative output method that allows Blender to serve frames over a network, useful for using external video encoders where the frames would not fit uncompressed on disk. documentation

VSE Rendering

Rendering to an Image Sequence

In many cases, cutting and re-arranging (editing) a codec-encoded video strip will give you fits, because the encoding algorithm that is used internally to reconstruct each image gets 'off' by a frame or two or three. To work directly on the 'raw' frame set, a very common technique is to import your video as a strip and render it out to series of individual frames, where each frame is stored in its own image file (JPG most commonly).

To do so, Add->Movie and load your original video. Set your Format SizeX and SizeY (either to match the original, or different if you want to distort or upscale/downscale the video), set image type to JPEG, adjust your Quality settings, and in the Anim panel set your End: to the number of actual frames in the video strip. Click ANIMATION and a series of numbered files will be output to the top filespec in the Output panel.

You can now delete the video strip, and Add->Image instead; right click on the directory name to pull in all of the images, in sequence, that are within that directory. Now, when you cut at frame 4321, for example, the next frame of the second strip will *really* start with frame 4322.

Rendering to Video

Ridiculously easy (when you learn where the buttons are):

1. Add the sequence of images as described above.
2. Set your Output file path and name to wherever you want to save the movie file (e.g. C:\My Documents\MyMovie) in the upper output box of the render buttons.
3. Change your Format to a movie file format (AVI, MOV, FFMPEG) and CODEC.
4. Set your framerate to match whatever framerate the sequence is to be played back in. Under the Anim/Playback buttons.
5. Set your ANIM End: to the number of images in the sequence, and
6. ANIM

The single movie file is created and saved; the name is what you specified but with the starting frame and ending frame numbers appended (e.g. MyMovie0000-0250.avi)

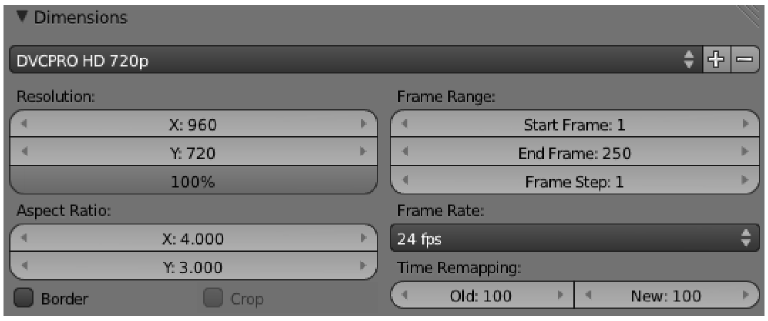
Preparing your work for video

Once you have mastered the trick of animation you will surely start to produce wonderful animations, encoded with your favorite codecs, and possibly you'll share them on the Internet with the rest of the community.

Sooner or later you will be struck with the desire to build an animation for television, or maybe burn your own DVDs. To spare you some disappointment, here are some tips specifically targeted at Video preparation. The first and principal one is to remember the double-dashed white lines in the camera view!

If you render for PC then the whole rendered image which lies within the *outer* dashed rectangle will be shown. For television, some lines and some part of the lines will be lost due to the mechanics of the electron beam scanning in your TV's cathode ray tube. You are guaranteed that what is within the *inner* dashed rectangle in camera view will be visible on the screen. Everything within the two rectangles may or may not be visible, depending on the given TV set that your audience watches the video on.

Dimensions Presets


▼ Dimensions

DVCPRO HD 720p

Resolution:
X: 960
Y: 720
100%

Frame Range:
Start Frame: 1
End Frame: 250
Frame Step: 1

Aspect Ratio:
X: 4.000
Y: 3.000

Frame Rate:
24 fps

Time Remapping:
Old: 100
New: 100

Border
Crop

The rendering size is strictly dictated by the TV standard. Blender has 11 pre-set settings for your convenience:

Preset	Resolution (X x Y)	Aspect Ratio (X x Y)	Frame Rate
DVCPRO HD 1080p	1280x1080	3:2	24 fps
DVCPRO HD 720p	960x720	4:3	24 fps
HDTV 1080p	1920x1080	1:1	24 fps
HDTV 720p	1280x720	1:1	24 fps
HDV 1080p	1440x1080	4:3	23.98 fps
HDV NTSC 1080p	1440x1080	4:3	29.97 fps
HDV PAL 1080p	1440x1080	4:3	25 fps
TV NTSC 16:9	720x480	40:33	29.97 fps
TV NTSC 4:3	720x486	10:11	29.97 fps
TV PAL 16:9	720x576	16:11	25 fps
TV PAL 4:3	720x576	12:11	25 fps

Note that if you render your animation at 1600x1200 resolution, and then burn a DVD, your image will not be clearer or crisper on the TV; in fact the DVD burning software will have had to downsize your images to fit the resolutions shown above, and you will have wasted about 4x disk space and render time.

Pixel Aspect Ratio

Older TV screens do *not* have the square pixels which Computer monitors have; their pixels are somewhat rectangular, so it is necessary to generate *pre-distorted* images which will look bad on a computer but which will display nicely on a TV set. It is important that you use the correct pixel aspect ratio when rendering to prevent re-sampling, resulting in lowered image quality.

Colour Saturation

Most video tapes and video signals are not based on the RGB model but on the YCrCb model: more precisely, the YUV in Europe (PAL), and the YIQ in the USA (NTSC), the latter being quite similar to the former. Hence some knowledge of this is necessary too.

The YCrCb model sends information as 'Luminance', or intensity (Y) and two 'Crominance' signals, red and blue (Cr and Cb). Actually a Black and White TV set shows only luminance, while color TV sets reconstruct color from Crominances (and from luminance). Construction of the YCrCb values from the RGB ones takes two steps (the constants *in italics* depend on the system: PAL or NTSC):

First, the Gamma correction (*g* varies: 2.2 for NTSC, 2.8 for PAL):

- $R' = R^{1/g}$:*$G' = G^{1/g}$
- $B' = B^{1/g}$

Then, the conversion itself:

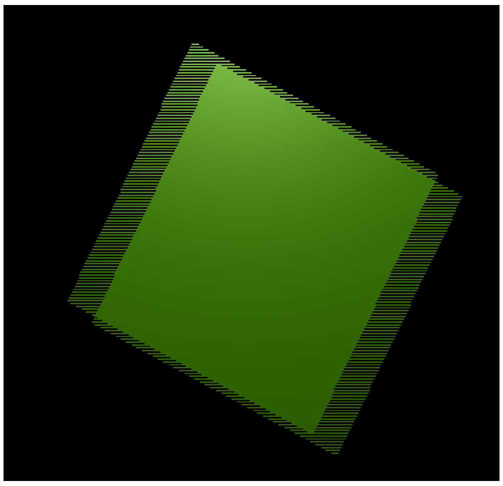
- $Y = 0.299R' + 0.587G' + 0.114B'$
- $Cr = a_1(R' - Y) + b_1(B' - Y)$
- $Cb = a_2(R' - Y) + b_2(B' - Y)$

Whereas a standard 24 bit RGB picture has 8 bits for each channel, to keep bandwidth down, and considering that the human eye is more sensitive to luminance than to chrominance, the luminance signal is sent with more bits than the two chrominance signals. This bit expansion results in a smaller dynamic of colors in video, than what you are used to on monitors. You hence have to keep in mind that not all colors can be correctly displayed.

A rule of thumb is to keep the colors as 'grayish' or 'unsaturated' as possible; this roughly means keeping the dynamics of your colors within 80% of one another. In other words, the difference between the highest RGB value and the lowest RGB value should not exceed 0.8 ([0-1] range) or 200 ([0-255] range).

This is not strict—something more than 0.8 is acceptable—but an RGB display with color contrast that ranges from 0.0 to 1.0 will appear to be very ugly (over-saturated) on video, while appearing bright and dynamic on a computer monitor.

Rendering to fields


Field Rendering result.

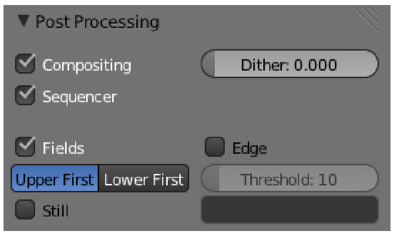
The TV standards prescribe that there should be 25 frames per second (PAL) or 30 frames per second (NTSC). Since the phosphors of the screen do not maintain luminosity for very long, this could produce a noticeable flickering.

To minimize this, a TV does not represent frames as a Computer does ('progressive' mode), but rather represents half-frames, or *fields* at a double refresh rate, hence 50 half frames per second on PAL and 60 half frames per second on NTSC. This was originally bound to the frequency of power lines in Europe (50Hz) and the US (60Hz).

In particular, fields are "interlaced" in the sense that one field presents all the even lines of the complete frame and the subsequent field the odd ones.

Since there is a non-negligible time difference between each field (1/50 or 1/60 of a second) merely rendering a frame the usual way and splitting it into two half frames does not work. A noticeable jitter of the edges of moving objects would be present.

Options



Field Rendering setup.

Fields

Enable field rendering. When the Fields button in the Render Panel is pressed (*Post Processing* section), Blender prepares each frame in two passes. On the first it renders only the even lines, then it *advances in time by half a time step* and renders all the odd lines. This produces odd results on a PC screen (*Field Rendering result.*) but will show correctly on a TV set.

Upper First / Lower First
: Toggles between rendering the even and odd frames first.

Still
: Disables the half-frame time step between fields (x).

Setting up the correct field order

Blender's default setting is to produce Even fields *before* Odd fields; this complies with European PAL standards. Odd fields are scanned first on NTSC.

Of course, if you make the wrong selection things are even worse than if no Field rendering at all was used!

If you are really confused, a simple trick to determine the correct field order is to render a short test animation of a white square moving from left to right on a black background. Prepare one version with odd field order and another with even field order, and look at them on a television screen. The one with the right field order will look smooth and the other one horrible. Doing this simple test will save you *hours* of wasted rendering time…

Fields and Composite Nodes

Nodes are currently not field-aware. This is partly due to the fact that in fields, too much information is missing to do good neighborhood operations (blur, vector blur etc.). The solution is to render your animation at double the frame rate without fields and do the interlacing of the footage afterwards.

Video Files

These formats are primarily used for compressing rendered sequences into a playable movie (they can also be used to make plain audio files).

A codec is a little routine that compresses the video so that it will fit on a DVD, or be able to be streamed out over the Internet, over a cable, or just be a reasonable file size. Codecs compress the channels of a video down to save space and enable continuous playback. *Lossy* codecs make smaller files at the expense of image quality. Some codecs, like H.264, are great for larger images. Codecs are used to encode and decode the movie, and so must be present on both the encoding machine (Blender) and the target machine. The results of the encoding are stored in a container file.

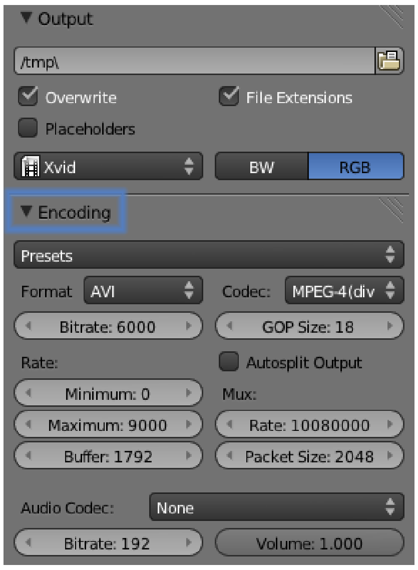
There are dozens, if not hundreds, of codecs, including XviD, H.264, DivX, Microsoft, and so on. Each has advantages and disadvantages and compatibility with different players on different operating systems.

Most codecs can only compress the RGB or YUV color space, but some support the Alpha channel as well. Codecs that support RGBA include:

- animation (quicktime)
- PNG *TIFF *Pixlet - not loss-less, and may be only available on Apple Mac.
- Lagarith Loss-less Video Codec

AVI Codec	AVI codec compression. Available codecs are operating-system dependent. When an AVI codec is initially chosen, the codec dialog is automatically launched. The codec can be changed directly using the Set Codec button which appears (*AVI Codec settings.*).
AVI Jpeg	AVI but with Jpeg compression. Lossy, smaller files but not as small as you can get with a Codec compression algorithm. Jpeg compression is also the one used in the DV format used in digital camcorders.
AVI Raw	Audio-Video Interlaced (AVI) uncompressed frames.
Frameserver	Blender puts out frames upon request as part of a render farm. The port number is specified in the OpenGL User Preferences panel.
H.264	Encodes movies with the H.264 codec. See Advanced Encoding.
MPEG	Encodes movies with the MPEG codec. See Advanced Encoding.
Ogg Theora	Encodes movies with the Theora codec as Ogg files. See Advanced Encoding.
QuickTime	Apple's Quicktime .mov file. The Quicktime codec dialog is available when this codec is installed and this format is initially chosen. See Quicktime Encoding. Reads GIF if QuickTime is Installed Blender can read GIF files on Windows and Mac platforms with [QuickTime] installed. The GIF capabilities (as well as flattened PSD, flattened PDF on Mac, and others) come along with QuickTime.
Xvid	Encodes movies with the Xvid codec. See Advanced Encoding.

Advanced Encoding



If the H.264, MPEG, Ogg Theora, or Xvid codecs are chosen, an Encoding panel becomes available. This has settings for encoding these file types, and other formats using FFmpeg.

FFmpeg, short for Fast Forward Moving Pictures Expert Group, is a collection of free and open source software libraries that can record, convert and stream digital audio and video in numerous formats. It includes libavcodec, an audio/video codec library used by several other projects, and libavformat, an audio/video container mux and demux library.

Video Settings

Here you choose which video codec you want to use, and compression settings. With all of these compression choices, there is a tradeoff between file size, compatibility across platforms, and playback quality.

When you view the System Console, you can see some of the output of the encoding process. You will see even more output if you execute Blender as *blender -d* .

You can use the presets, DV, SVCD, DVD, etc. which choose optimum settings for you for that type of output, or you can manually select the format (MPEG-1, MPEG-2, MPEG-4, AVI, Quicktime (if installed), DV, H.264, or Xvid (if installed). You must have the proper codec installed on your computer for Blender to be able to call it and use it to compress the video stream.

Video Formats

Name	Extensions	Description
MPEG-1	.mpg, .mpeg	A standard for lossy compression of video and audio. It is designed to compress VHS-quality raw digital video and CD audio down to 1.5 Mbit/s.
MPEG-2	.dvd, .vob, .mpg., .mpeg	A standard for "the generic coding of moving pictures and associated audio information". It describes a combination of lossy video compression and lossy audio data compression methods which permit storage and transmission of movies using currently available storage media and transmission bandwidth.
MPEG-4(DivX)	.mp4, .mpg, .mpeg	Absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, and adds new features.
AVI	.avi	A derivative of the Resource Interchange File Format (RIFF), which divides a file's data into blocks, or "chunks."
Quicktime	.mov	A multi-tracked format. QuickTime and MP4 container formats can use the same MPEG-4 formats; they are mostly interchangeable in a QuickTime-only environment. MP4, being an international standard, has more support.
DV	.dv	An intraframe video compression scheme, which uses the discrete cosine transform (DCT) to compress video on a frame-by-frame basis. Audio is stored uncompressed.
H.264	.avi ("for now")	A standard for video compression, and is currently one of the most commonly used formats for the recording, compression, and distribution of high definition video.
Xvid	.avi ("for now")	A video codec library following the MPEG-4 standard. It uses ASP features such as b-frames, global and quarter pixel motion compensation, lumi masking, trellis quantization, and H.263, MPEG and custom quantization matrices. Xvid is a primary competitor of the DivX Pro Codec.
Ogg	.ogg, .ogv	A free lossy video compression format. It is developed by the Xiph.Org Foundation and distributed without licensing fees.
Matroska	.mkv	An open standard free container format, a file format that can hold an unlimited number of video, audio, picture or subtitle tracks in one file.
Flash	.flv	A container file format used to deliver video over the Internet using Adobe Flash Player.
Wav	.wav	An uncompressed (or lightly compressed) Microsoft and IBM audio file format.
Mp3	.mp3	A highly-compressed, patented digital audio encoding format using a form of lossy data compression. It is a common audio format for consumer audio storage, as well as a de facto standard of digital audio compression for the transfer and playback of music on digital audio players.

Video Codecs

Name	Description
None	*For audio-only encoding.*
MPEG-1	(See Video Formats, above.)
MPEG-2	(See Video Formats, above.)
MPEG-4(DivX)	(See Video Formats, above.)
HuffYUV	Loss-less video codec created by Ben Rudiak-Gould which is meant to replace uncompressed YCbCr as a video capture format.
DV	(See Video Formats, above.)
H.264	(See Video Formats, above.)
Xvid	(See Video Formats, above.)
Theora	(See Ogg in Video Formats, above.)
Flash Video	(See Video Formats, above.)
FFmpeg video codec #1	A.K.A. FFV1, a loss-less intra-frame video codec. It can use either variable length coding or arithmetic coding for entropy coding. The encoder and decoder are part of the free, open-source library libavcodec in FFmpeg.

Options

Bitrate
: Set the average bitrate (quality), which is the count of binary digits per frame. See also: ffmpeg -b:v

Rate
: The bitrate control also includes a Minimum and a Maximum.

  Buffer

  : The decoder bitstream buffer size.

GOP Size
: The number of pictures per Group of Pictures. Set to 0 for "intra_only", which disables inter-frame video. From ffmpeg docs: "For streaming at very low bitrate application, use a low frame rate and a small GOP size. This is especially true for RealVideo where the Linux player does not seem to be very fast, so it can miss frames"

Autosplit Output
: If your video is HUGE and exceeds 2Gig, enable Autosplit Output. The main control over output filesize is the GOP, or keyframe interlace. A higher number generally leads to a smaller file, but needs a higher-powered device to replay it.

Mux
: Multiplexing settings.

  Rate

  : Maximum bit rate of the multiplexed stream.

  Packet Size

  : (Undocumented in ffmpeg)

Standards
Codecs cannot encode off-the-wall video sizes, so stick to the XY sizes used in the presets for standard TV sizes.

Audio Settings

Audio is encoded using the codec you choose.

Audio Codecs

MP2	A lossy audio compression format defined by ISO/IEC 11172-3.
MP3	(See MP3 in Video Formats, above.)
AC3	Audio Codec 3, an audio compression technology developed by Dolby Laboratories.
AAC	"Advanced Audio Codec," a standardized, lossy compression and encoding scheme for digital audio. Designed to be the successor of the MP3 format, AAC generally achieves better sound quality than MP3 at similar bit rates.
Vorbis	An open-standard, highly-compressed format comparable to MP3 or AAC. Had been shown to perform significantly better than many other lossy audio formats in the past in that it produced smaller files at equivalent or higher quality while retaining computational complexity comparable to other MDCT formats such as AAC or Windows Media Audio.
FLAC	Free Loss-less Audio Codec. Digital audio compressed by FLAC's algorithm can typically be reduced to 50–60% of its original size, and decompressed into an identical copy of the original audio data.
PCM	Pulse Code Modulation, a method used to digitally represent sampled analog signals. It is the standard form for digital audio in computers and various Blu-ray, Compact Disc and DVD formats, as well as other uses such as digital telephone systems

Bitrate
: For each codec, you can to control the bitrate (quality) of the sound in the movie. This example shows MP3 encoding at 128kbps. Higher bitrates are bigger files that stream worse but sound better. Stick to powers of 2 for compatibility.

Samplerate
: Samplerate controls the number of samples per second of the audio. The default, 44100, is standard for many file types, including CD audio, and produces a high quality sound.

Volume
: Set the output volume of the audio.

Tips

Choosing which format to use depends on what you are going to do with the image.

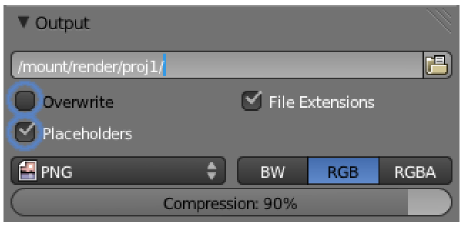
If you are animating a movie and are not going to do any post-processing or special effects on it, use either **AVI-JPEG** or **AVI Codec** and choose the XviD open codec. If you want to output your movie with sound that you have loaded into the VSE, use **FFMPEG**.

If you are going to do post-processing on your movie, it is best to use a frame set rendered as **OpenEXR** images; if you only want one file, then choose **AVI Raw**. While AVI Raw is huge, it preserves the exact quality of output for post-processing. After post-processing (compositing and/or sequencing), you can compress it down. You don't want to post-process a compressed file, because the compression artifacts might throw off what you are trying to accomplish with the post-processing.

Note that you might not want to render directly to a video format. If a problem occurs while rendering, you have to re-render all frames from the beginning. If you first render out a set of static images (such as the default PNG, or the higher-quality OpenEXR), you can stitch them together with an Image Strip in the Video Sequence Editor (VSE). This way, you can easily:

- Restart the rendering from the place (the frame) where the problem occurred.
- Try out different video options in seconds, rather than minutes or hours.
- Enjoy the rest of the features of the VSE, such as adding Image Strips from previous renders, audio, video clips, etc.

Home-made Render Farm



/mount/render/proj1/
Overwrite
File Extensions ✓
Placeholders ✓
Compression: 90%

An easy way to get multiple machines to share the rendering workload is to:

1. Set up a shared directory (such as a Windows Share or an NFS mount)
2. Un-check "Overwrite" and check "Placeholders"
3. Start as many machines as you wish rendering to that directory -- they will not step on each other's toes.

Post Processed Effects

There are several effects you can enable in the Render Settings that add visual elements to rendered images, after the rendering has completed. These are not done in camera, but rather composited on top of the image.

Composited and Sequence are discussed in Output Options.

Fields are discussed in Video Output.

Render Layers

Render layers are used to separate your composite image into layers. Use Render Layers for a specific reason - such as creating depth of field, relighting isolated elements within the image via a normal pass, adding a colorcast to specific portions of the image, etc. The keyword here is isolation. Render layers allow you to dissect, effect and/or correct individual elements or groups within your composition before outputting your final render. This saves you from endlessly re-rendering your scene just to find out whether a correction is going to work or not.

Render Layers in Compositing

What are Render Layers *really* used for? Blender's node-based compositing system!

In Nodes, when you add an Input Node of type *RenderLayers,* and select the Scene, you bring in whatever information you've specified for that RenderLayer. This node becomes a source for the rendering pipeline products you've specified *(see below)*, as applied to the objects in the qualifying layer(s). Each of these products then "flow out of" that node toward their appointed destinations in the node graph you've constructed.

Layers or Passes?

Blender's Render Pass system is a subset of Render Layers. Passes are specific to elements of shading properties, such as specular and diffuse, which can later be combined in compositing. Render Layers are more geared for separating scene components, but can include isolated passes as well.

Using Render Layers

In Render buttons, open the Layers tab. This is where you select the layers that you want to render, and the settings for the upcoming render.

Enabling and Naming

The list box contains the Render Layers that you have created, and options for disabling, removing, adding, and renaming layers.

Note that the settings in the Layers tab are Render Layer specific. Make sure that you have the appropriate Render Layer selected when changing settings.

The checkbox enables or disables the computation of the whole render layer. Enable only those layers you are working with to save time. The selector allows you to scroll through and examine existing render layers, or to add a new one.

Creating a new Render Layer

By default, there is 1 Render Layer created for you, and it includes all layers, whether they are used in your scene or not. To add yet another Render Layer, click the yellow up-down selector and select Add New render layer. You now have two Render Layers to choose from, and the active one is shown in the window. Each Render Layer will have its own set of layers that are rendered (sort of makes sense now, doesn't it?).

For example, you might have a robot in a scene with a ground object, buildings, etc. If the robot is on visible-layer 5, you can create one render layer named "Robot" with layer 5 selected in both the Scene: and Layer: buttons.

You can create another render layer (maybe named "stuff") that has all other layers EXCEPT layer 5 selected in both the Scene: and Layer: buttons. Then, back in the Node Editor, you create TWO input nodes of type Render Layer: one for the Robot Render Layer, and another for the other Stuff. Run both through a mixer and out to the Composite viewer to get the big picture.

Scene Layers Settings

There are three sets of scene layer buttons:

Scene
: These mirror the layer buttons in the 3d view header, and tell which scene layers are visible when rendering.

Layer
: Control which scene layers are included in the current Render Layer.

Exclude
: Exclude render layer so it will not influence to another layers.

Mask Layers
: The image rendered is from the objects that are between the selected layer(s) and the Z-mask layers. In the example, the cube is on layers 2 and 3, and the grass in on layer 1. In the render layer which we have arbitrarily chosen to call "zmask", as shown in the picture above, layer 1 is selected and layer 3 is designated as the Z-mask (as indicated by the black dot). Therefore, only that part of Layer 1 which is in front of the object on layer 3 (the cube) is rendered.

You can select that layer by LMB clicking the button. To select multiple layers, ⇧ Shift LMB click. (The dot in the button in this case turns *dark gray.*)

Layer Sets AND each other
Only the objects in layers that are selected BOTH in the main Scene Layer group AND the Render Layer Layer group will be rendered. So, if the Scene has only Layer 1 selected, and your Render Layer set specifies to render only Layers 2 and 3, nothing but the Sky (if selected) will be rendered.

Overrides

The Light and Material selector boxes allow you to override materials and lights per layer, applying them to all objects in the Render Layer.

Light
: Enter the name of a light group, and the scene will be lit with only those lights. Usually, you use this to speed up draft renders of a scene that has complicated lighting, by entering the name of a small group of key lights.

Material
: Overrides all material settings to use the name of the Material entered. Use this to speed up draft renders. Use the default material to check basic lighting.

Include Options

Each render layer has its own set of major products to include in the rendering pipeline. To save time and give you control when working with passes, this set of buttons allow you to select which major products to render:

Z-mask
: Only render what's in front of the solid z values.
: Negate

    Only render what's Behind the solid z values.

AllZ
: Z-values are computed for everything in view, not just those things that are rendered. When disabled, objects not included in the render have no ("infinite") z value.

Solid
: Solid faces are rendered. All normal meshes are solid faced.

Halo
: Halo materials are rendered.

Z-transp
: Transparency may be Z-based or Ray-traced. If Z-based, enabling *Ztra* renders transparent areas with the z-value of what is behind the transparent area.

Sky
: Turning on Sky renders the sky, as defined in your material world settings. Otherwise, a black alpha transparent background is rendered.

Edge
: If Edge is enable in the Output panel, objects in this Render Layer are given an outline edge. Turning on Edge pulls in the Edge settings from the Output tab, and adds an outline to the objects. Edges also have to be enabled on the Output tab.

Strand
: Strands are strings of static particles that are colored as part of the material settings; they look like strands of hair or grass.

Passes

Render Passes (Combined, Z, Vec, etc.) are discussed on the next page.

Examples

Rendering only certain objects

For example, suppose you have added a cool halo to your robot and you want to quickly see what it looks like. Suppose your scene has boxes on layer 1, laser rifles on layer 2, the robot on layer 5, and lights and camera on layer 20, and they are all selected and visible in the 3d view. If you want to render just your robot, and he is on layer 5, you click on the render layer 5 button (which is below the Render Layer name), de-select sky (so that the sky/horizon is not rendered) and select Halo. Presto! When you render, only the robot is rendered (quickly) and not all the other elements of your scene (like the boxes he is running in front of).

Outlining only selected objects

To render an image where only one or two of the objects are outlined, move those objects onto layer(s) separate from everything else. Create Render Layer 1 for those layer(s) by selecting only those layers in the Render Layer layer set. Create Render Layer 2 for the other stuff. Enable the Edge option for Render Layer 1 (remember to also enable Edge on the Output tab) and make sure it is de-selected (off) for Render Layer 2. In the Node Editor, create two input nodes, one for each Render Layer. Mix the two images. Done. Simple. Yea.

Render Passes

Render Passes are necessary because of the different things the Blender Render Engine must calculate to give you the final image. In each 'pass' the engine calculates different interactions between objects.

Render Passes In Detail

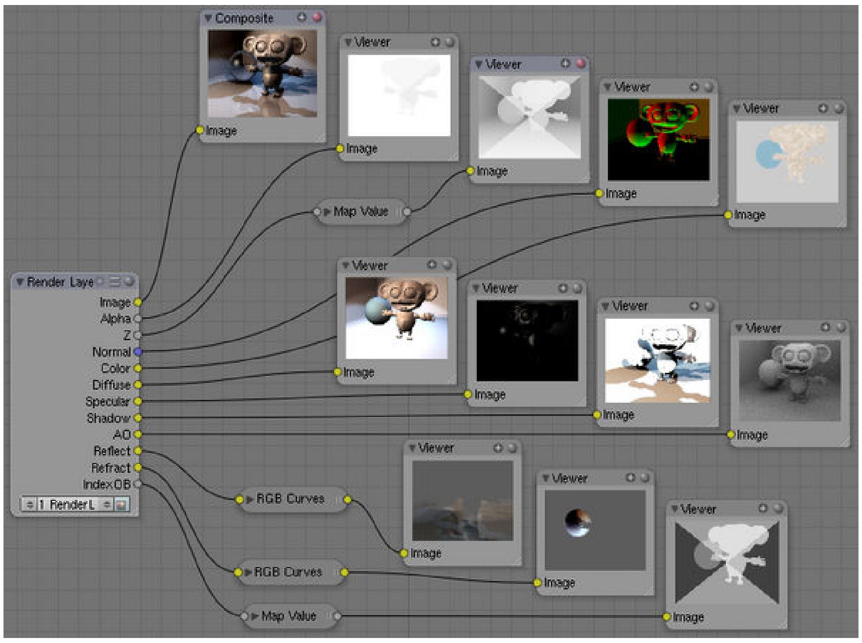
Everything you see in a render must be calculated for the final image. All interactions between objects in your scene, lighting, cameras, background images, world settings, etc., must all be separately calculated in different passes for different reasons, such as calculating shadows.

In a render, every pixel has been calculated several times to make sure it will show the right color for the right part of the image. Various things that are calculated in a standard render include:

- *Where are **shadows** cast?*
- *How is **ambient** light in the environment blocked (**occluded**) by objects in the scene?*
- *How is light **reflected** off mirrored surfaces?* Like shadows, lines are calculated, except this time they come from the camera and bounce off mirrored surfaces, so that when these lines hit an object, the engine calculates that this is what the camera should see.
- *How is light bent (**refracted**) as is passes through transparent objects?* Does it go straight through? Does it bend? If so, at what depth in the object?
- *What designated **objects** are in the scene, and what is their outline?* Should the object appear blurred, or should it appear in sharp focus?
- *How fast is something moving (**velocity**)?* Should it appear blurred given our frame rate or is it slow enough to still be focused on properly?
- *How far away from the camera are objects' surfaces (**Z-depth**)?* Can the object's surfaces be seen at all, or are they being blocked by another object's geometry?
- *Does an object have a **normal** vector (bumpmap)?* Do shadows and apparent geometry need to be calculated for any objects?
- *Is there any **specularity**?* Are objects with textures such as metal shiny at all?

Renderer Rewrite
Starting with Blender 2.42, the render engine was rewritten. See Unified Renderer if you are using an old version of Blender.

The answer to each of the above questions is an image or map, as shown below:

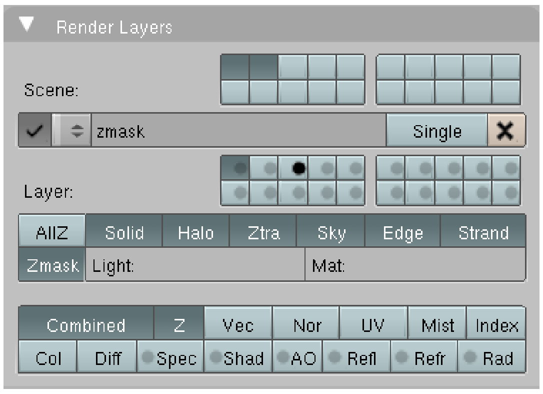


Each Render Pass puts out an image or a map. For the purposes of this example, a Render Layer was defined to produce all possible outputs. When a Render Layer input-node was added to the node diagram and the Render Layer input-node was subsequently associated with the Render Layer, all of the layer's outputs appeared as connection points on the right-hand (output) side of the node.

Render Passes that produce Images can be directly viewed in a viewer, or, if they are the only pass that is rendered, saved as the render image. If the pass is enabled, it can be saved in a multilayer OpenEXR format.

If the Render Pass output is not an image but is a map, it needs to be translated into something that we can see. For example, the Z-depth map is an array of values that specifies how far away from the camera each pixel is; values range between +/-3,000,000 Blender Units or so. The intermediate node you see above, between the Render Layer output socket and the Viewer node input socket (such as Map Value) does this translation or scaling. You must use that specific kind of translation node to get good results if you intend on operating on that map as an image. You must then, after making any adjustments, run the map back through that node to re-scale it back to the original before saving.

Selecting Render Passes



Render Passes are the various distinct outputs that the renderer is able to generate. All of the following render outputs are normally combined into a single output known, appropriately enough, as the **Combined** output. But you can also select any of them to be output as a separate pass. (If you do so, in most cases you can choose whether to *also* continue to include it in the Combined output.)

Some of these outputs must be enabled and used within your scene (and not just selected in the Render Layer panel) in order to show anything. For example, if you do not have any lights in your scene, or those lights have been set to not cast shadows, or objects in the limelight do not have materials which have been set to receive shadows, the **Shadow** pass will be blank; there's simply nothing to show you. If you have not enabled *Ambient Occlusion* in your World environment settings, the **AO** pass will be blank, even if you select it here.

To save time and disk space, you have to tell Blender each of the passes to render in the Render Layers panel (which we first introduced on the previous page):

Combined	This renders everything in the image, even if it's not necessary. ("The whole enchilada," so to speak.) This is all the options below, blended into a single output, *except* those options which you've indicated should be omitted from this pass, as indicated with the camera button.
Z	The Z-depth map; how far away each pixel is from the camera. Used for Depth-Of-Field (DOF). The depth map is inverse linear *(1/distance)* from the camera clip start.
Vector	The direction and speed things are moving. Used with Vector Blur.
Normal	Calculates lighting and apparent geometry for a bumpmap (an image which is used to fake detail on an object) or for changing the apparent direction of light falling on an object.
UV	Allows texturing after rendering. See UV node.
Mist	Deliver Mist factor pass.
Object Index	Masks selected objects. See MaskObj node.
Color	The color of materials without shading.
Diffuse	The diffuse shading of materials.
Specular	Specular highlights.
Shadow	Shadows cast. Make sure shadows are cast by your lights (positive or negative), and received by materials. To use this pass, mix multiply it with the Diffuse pass.
Emit	Emission pass.
AO	Ambient Occlusion. Make sure it's turned on in your environment and that RayTracing is enabled.
Environment	Environment lighting.
Indirect	Indirect lighting pass.
Reflection	Reflection off mirrors and other reflective surfaces (highly waxed white floors, for example). Mix Add this pass to Diffuse to use it.
Refraction	Refraction of colors through transparent meshes. Mix Add this pass to the Diffuse pass to use it.

When you enable a pass, the appropriate socket on the Render Layers node shows up like magic, and can be used as shown in the example above.

Excluding Render Passes

As we said, the **Combined** output is an amalgam of several outputs which are *also* available separately. When you select one of these outputs, they will be provided separately *and also* included in the Combined pass.

When you enable the Camera icon that is beside several of the pass options, the particular pass will be excluded from the combined pass. They will be made available separately *but not* included in the combined pass.

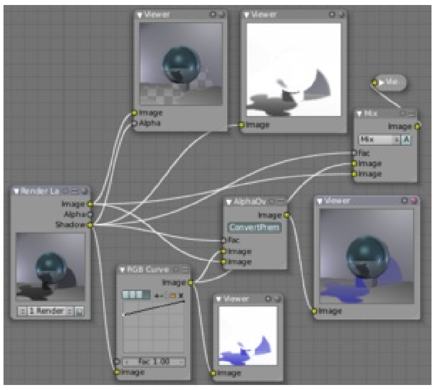
Using Render Passes

The primary purpose of Render Passes is to enable you to process the various outputs in different ways, by constructing networks of render nodes. You can achieve many special effects, and economize considerably on the render times of complicated scenes, by creative and effective use of this facility. We'll show you a few examples of this in just a moment.

Quite a bit of information about the typical uses for some of the passes is discussed elsewhere:

- Image: Since this is the main product, all of Blender uses it.
- Alpha: See the *AlphaOver* node and all of the *Matte* nodes.
- Z: See the *Defocus* node.
- Vec: See the *Vector Blur* node.
- Normal: See the *Normal* node.

Recoloring Shadows



Let's run the Shadow buffer through a colorization noodle, then recombine it; all your shadows will be artificially colored. Lots of threads in this noodle are shown to the right, so let's walk through it. On the left is the Render Layer input node: it refers to one of the Render Layers that we have defined for our scene. In the scene, we have a reflective ball on a pedestal standing in front of a backdrop. Everything (except the ball) is gray. We use a standard four-light rig: backfill placed high, two sidefills at ground level, and a key light above and to the left of camera. Suzanne, a monkey-shaped geometry, is standing in front of the key light, so her shadow is cast into the scene on the floor. The ball casts shadows onto the backdrop and floor.

The output channels of the Render Layer node are determined by which buttons we selected when defining our Render Layer. The top two viewers show you the image output using the Shadow as the Alpha channel, and the node next to it just the Shadow channel. Where the Shadow is dark, the image in the left viewer is transparent. We have used the Shadow to cut out parts of the image.

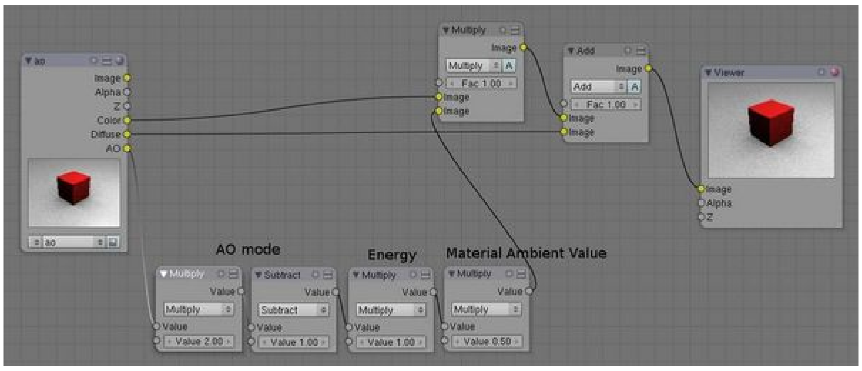
We then take the shadow through an RGB Curve, which is set to magnify just the Blue by 75%; so a gray shadow of (R:40, G:40, B:40) becomes (R:40, G:40, B:40x1.75=70). That blue-tinged shadow is shown in the bottom viewer. Now we have two options: AlphaOver and Mix. For either option:

- Use the Shadow map as a Factor.
- Feed the Blue Shadow to the Top Socket.
- Feed the core or base image to the Bottom Socket.

The resulting image is the same in either case; a blue shadow. Note that Suzanne's reflection is not blue; there's a different Render Pass for that.

You could just as easily swap in another image entirely; for example, the shadow map from another render layer. You can even take an image from another project entirely and use that instead (using the Image Input node), to get a different effect. (For example, an effect similar to a *Star Wars Episode One* movie poster, where Anakin Skywalker already casts the shadow of Darth Vader.)

Compositing Ambient Occlusion



AO is a geometry-based dirt shader, making corners darker. It is separately enabled in the World settings and computed as a separate pass. When enabled, it has one of three Modes (*Add, Subtract, Both*), and variable *Energy* level (which changes the intensity of the shading). The third variable is the amount of Ambient light that the material receives. If it does not receive any, then ambient occlusion does not affect it. Based on these variables, Blender computes an AO pass. If you call it out as a separate pass and wish to composite it back into your image, you will need to enable the Color and Diffuse pass as well.

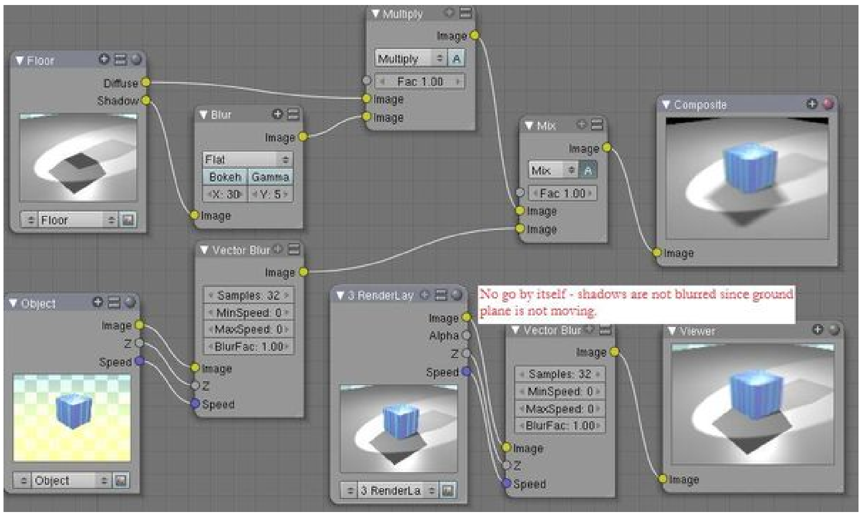
To configure your noodle, consider the example image above.

1. . First, depending on the AO mode do one of the following: If AO mode is Add: directly use the AO pass. If AO mode is Sub: Calculate AO - 1, or if AO mode is Both: Calculate 2*AO - 1.
2. . Multiply the output of Step 1 with the AO energy level.
3. . Multiply the output of Step 2 with the material's ambience value. If you have materials which receive different ambience light

levels (0.5 is the default), one would have to create an ambience map based on Object ID.
4. . Multiply the output of Step 3 with the color pass.
5. . Add the output of Step 4 to the diffuse pass.

If shadows, colored ambient light, specularity, reflections, and/or refractions are involved they have to be added to the diffuse pass before adding the converted AO pass.

Vector Blurring Shadows



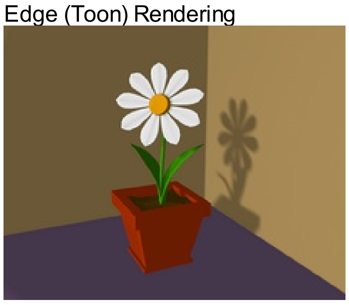
When using Vector Blur instead of Motion Blur, objects in motion are blurred, but objects at rest (with respect to the camera) are not blurred. The crossover is the shadow of the object in motion. Above, we have a cube in motion across a ground plane. If we just ran the combined pass through Vector Blur, you can see the result in the lower right-hand corner; the box is blurred, but its shadow is sharply in focus, and thus the image does not look realistic.

Therefore, we need to separate out the diffuse and shadow passes from the floor by creating a "Floor" render layer. That render layer has Diffuse and Shadow passes enabled, and only renders the floor object (layer 2). Another render layer ("Cube") renders the Z and Vector passes, and only renders the cube (on layer 1). Using the Blur node, we blur the shadow pass, and then combine the diffuse and blurred shadow by multiplying them together in a Mix Multiply node; we then have a blurred shadow on a crisp ground plane. We can then mix the vector-blurred object to provide a realistic-looking image.

Conclusion

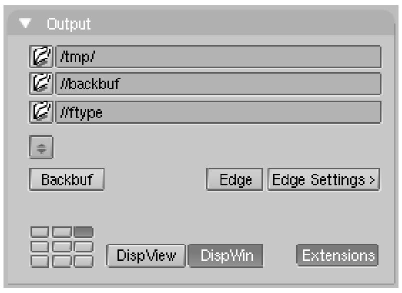
Render Passes can be manipulated to give you almost complete control over your final image. Causing objects to cast shadows that aren't really their shadows, making objects appear out of focus or sharply in focus like a real camera, manipulating colors just for final post-processing or just reconfiguring your render passes to save render time, are all things which you might wish to manipulate the render engine for.

Edge (Toon) Rendering


A scene with Toon materials.

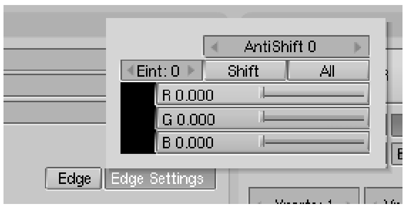
Blender's toon shaders can give your rendering a comic-book-like or manga-like appearance, affecting the shades of colors. The effect is not perfect since real comics and manga also usually have china ink outlines. Blender can add this feature as a post-processing operation.

Options



Toon edge buttons *(F10)*.

Edge
: This makes Blender search for edges in your rendering and add an 'outline' to them.



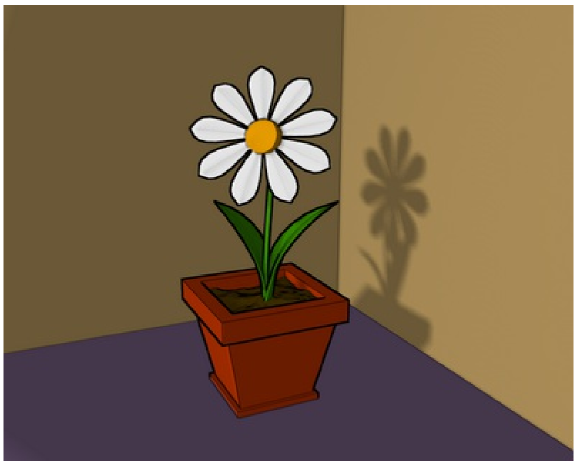
Toon edge settings *(F10)*.

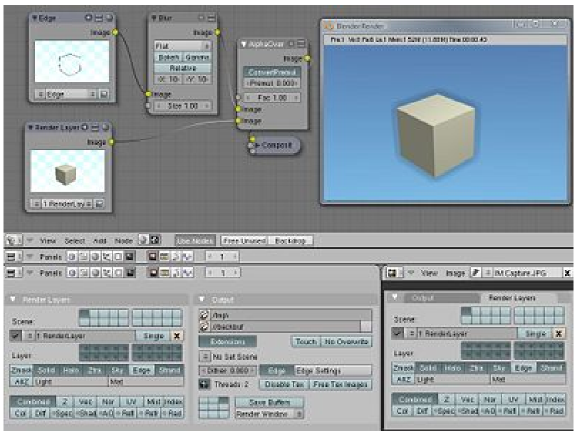
Before repeating the rendering it is necessary to set some parameters:

Threshold
: The threshold of the angle between faces for drawing edges, from 0 to 255. A value of 10 would just give outline of object against the background, whereas higher settings start to add outlines on surface edges, starting with sharper angles. At maximum intensity, Edge will even faintly display geometry subsurf edge lines in areas of imperfect smoothing.

Color / R/G/B
: The color of the rendered edges (black by default). Click on the swatch to see the color picker

Examples


Scene re-rendered with toon edge set.



It is possible to separate out the edge layer using a render layer dedicated to that purpose. The alpha channel is 0 where there is no edge, and 1 where the edge is. By separating out the edge layer, you can blur it, change its color, mask it, etc. The image to the right shows how to do this. I created an Edge render layer that only has the Sky and Edge layers (I included sky so that we get the world color later on in the composite output). The other render layer omits the Edge layer, so it returns just the normal image. On the output panel I enabled Edge with a width of 10 in black. I run that layer through a blur node. Using the Alphaover node, I then composite the cube on top of the blurred edge. The result gives a soft-shadow kind of effect. Note that Premultiply is set because the Edge image already has an alpha of 1.0 set.

Dithering

Dithering is a technique for blurring pixels to prevent banding that is seen in areas of gradients, where stair-stepping appears between colors. Banding artifacts are more noticeable when gradients are longer, or less steep. Dithering was developed for graphics with low bit depths, meaning they had a limited range of possible colors.

Dithering works by taking pixel values and comparing them with a threshold and neighboring pixels then does calculations to generate the appropriate color. Dithering creates the perceived effect of a larger color palette by creating a sort of visual color mixing. For example, if you take a grid and distribute red and yellow pixels evenly across it, the image would appear to be orange.

The Dither value ranges from 0 to 2.

Stamp

The stamp panel includes options for image stamping. Stamping adds text over the rendered image.

Note that stamp information is also written into the metadata of images that support it (currently JPEG, EXR, PNG), even when the stamp checkbox is disabled, metadata is written into the file for checked items.

Stamping can include the following data:

Time
: Include the current scene time and render frame as HH:MM:SS.FF.

Date
: Include the current date and time.

RenderTime
: Include the render time in the stamp image.

Frame
: Include the frame number.

Scene
: Include the name of the active scene.

Camera
: Include the name of the active camera.

Lens
: Include the name of the active camera's lens value.

Filename
: Include the filename of the .blend file.

Marker
: Include the name of the last marker.

Seq. Strip
: Include the name of the foreground sequence strip.

Note
: Include a custom note.

Stamp Text Color
: Set the color and alpha of the stamp text.

Stamp Background
: Set the color and alpha of the color behind the text.

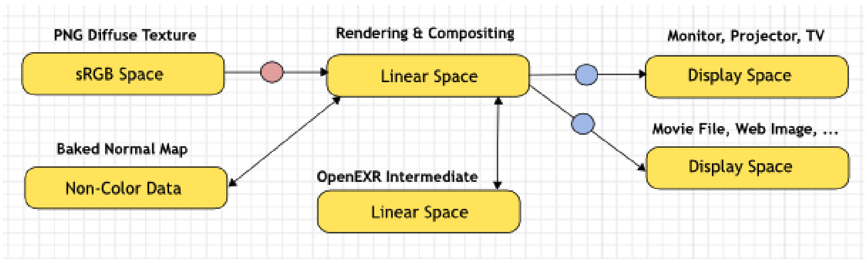
Font
: Set the size of the text.

Color Management

OpenColorIO is now integrated into Blender, along with a redesign of the color management system. Previously Blender only supported two color spaces, linear and sRGB; now many more are supported, with finer control over which color transformations should be used.

Scene Linear Color Space

For correct results, different color spaces are needed for rendering, display and storage of images. Rendering and compositing is best done in **scene linear color space**, which corresponds more closely to nature, and makes computations more physically accurate.



Example linear workflow

If the colors are linear, it means that if in reality we double the amount of photons, the color values are also doubled. Put another way, if we have two photos/renders each with one of two lights on, and add those images together, the result would be the same as a render/photo with both lights on. It follows that such a radiometrically linear space is best for photorealistic rendering and compositing.

However these values do not directly correspond to human perception or the way display devices work, and image files are often stored in different color spaces, so we have to take care to do the right conversion into and out of this linear color space.

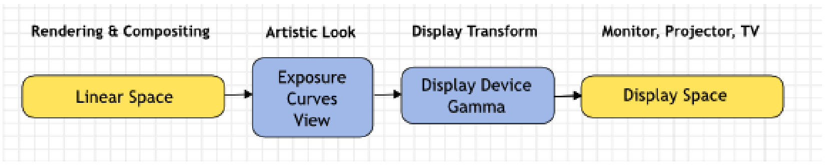
Scene Settings

These settings are found in the scene tab, under the Color Management panel.

Display

Correct display of renders requires a **conversion to the display device color space**, which can be configured here. A computer monitor works differently from a digital cinema project or HDTV. The scene properties have these settings:

Device
: The device that the image is being viewed on. Most computer monitors are configured for the sRGB color space, and so when working on a computer usually this option should just be left to the default. It would typically be changed when viewing the image on another display device connected to the computer, or when writing out image files intended to be displayed on another device. Rec709 is commonly used for HDTVs, while XYZ and DCI-P3 are common for digital projectors. Color management can be disabled by setting the device to None.

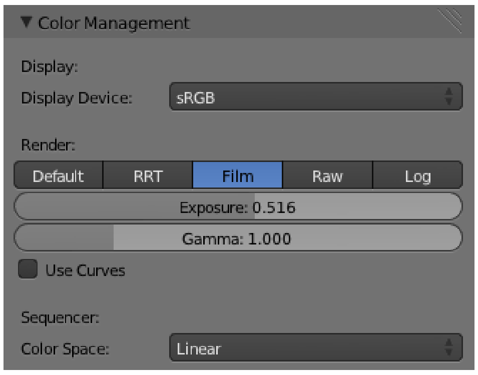


Conversion from linear to display device space

Render

Besides this, there is also an **artistic choice** to be made for renders. Partially that's because display devices can't display the full spectrum of colors and only have limited brightness, so we can squeeze the colors to fit in the gamut of the device. Besides that it can also be useful to give the renders a particular look, e.g. as if they have been printed on real film.

Another common use case is when you want to inspect renders, to see details in dark shadows or bright highlights, or identify render errors. Such settings would be only used temporarily and not get used for final renders.


▼ Color Management

Display:
Display Device: sRGB

Render:
Default | RRT | Film | Raw | Log
Exposure: 0.516
Gamma: 1.000
Use Curves

Sequencer:
Color Space: Linear

Scene settings for color management

View Transform
These are different ways to view the image on the same display device.

Default

Does no extra conversion besides the conversion for the display device.

RRT

Uses the ACES Reference Rendering Transform, to simulate a film-like look.

Film

This option is another film-like look.

Raw and Log

Intended for inspecting the image but not for final export. Raw gives the image without any color space conversion, while Log gives a more "flat" view of the image without very dark or light areas.

Exposure
Multiplier for the image brightness applied before color space conversion.

Gamma
Extra gamma correction applied after color space conversion. Note that the default sRGB or Rec709 color space conversions already include a gamma correction of approximately 2.2 (except the Raw and Log views), so this would be applied in addition to that.

RGB Curves
Curves to control image colors before color space conversion.
Color Unpremultiply
For premultiplied alpha render output, do color space conversion on colors without alpha, to avoid fringing on light backgrounds.

Sequencer

Sequencer Color Space
The color space that the sequencer operates in. By default the sequencer operates in sRGB space like it did in previous versions, but it can also be set to work in Linear space like the Compositing nodes, or another color space. Different color spaces will give different results for color correction, cross fades, and other operations.

Different views and exposures of the same render

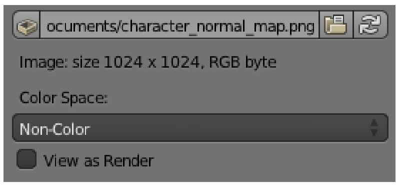
Image Files

The other place to keep color management in mind is when **loading and saving image files**. File formats such as PNG or JPEG will typically store colors in a color space ready for display, not in a linear space. When they are, for example, used as textures in renders, they need to be converted to linear first, and when saving renders for display on the web, they also need to be converted to a display space. Other file formats like OpenEXR store linear color spaces and as such are useful as intermediate files in production.

When working with image files, the default color space is usually the right one. If this is not the case, the color space of the image file can be configured in the image settings. A common situation where manual changes are needed is when working with or baking normal maps or displacement maps, for example. Such maps do not actually store colors, just data encoded as colors. In such cases they should be marked as Non-Color Data.

Image datablocks will always store float buffers in memory in the scene linear color space, while a byte buffer in memory and files on disk are stored in the color space specified with this setting:

Color Space
The color space of the image on disk. This depends on the file format, for example PNG or JPEG images are often stored in sRGB, while OpenEXR images are stored in a linear color space. Some images such as normal, bump or stencil maps do not strictly contain 'colors', and on such values no color space conversion should ever be applied. For such images the color space should be set to None.


ocuments/character_normal_map.png
Image: size 1024 x 1024, RGB byte
Color Space:
Non-Color
View as Render
Image settings for color management

By default only renders are displayed and saved with the render view transformations applied. These are the Render Result and Viewer image datablocks, and the files saved directly to disk with the Render Animation operator. However when loading a render saved to an intermediate OpenEXR file, Blender can't detect automatically that this is a render (it could be e.g. an image texture or displacement map). We need to specify that this is a render and that we want the transformations applied, with these two settings:

View as Render
: Display the image datablock (not only renders) with view transform, exposure, gamma, RGB curves applied. Useful for viewing rendered frames in linear OpenEXR files the same as when rendering them directly.

Save as Render
: Option in the image save operator to apply the view transform, exposure, gamma, RGB curves. This is useful for saving linear OpenEXR to e.g. PNG or JPEG files in display space.

World Settings

Settings in the World panel give you two additional controls for exposure, however, these bake the exposure effects into the rendered image, as opposed the techniques explained on this page, which affect the color space of rendered images.

See Exposure, for details.

OpenColorIO Configuration

Blender comes with a standard OpenColorIO configuration that contains a number of useful display devices and view transforms. The reference linear color space used is the linear color space with Rec. 709 chromaticities and D65 white point.

However OpenColorIO was also designed to give a consistent user experience across multiple applications, and for this a single shared configuration file can be used. Blender will use the standard OCIO environment variable to read an OpenColorIO configuration other than the default Blender one. More information about how to set up such a workflow can be found on the OpenColorIO website.

We currently use the following color space roles:

- *scene_linear*: color space used for rendering, compositing, and storing all float precision images in memory.
- *default_sequencer*: default color space for sequencer, *scene_linear* if not specified
- *default_byte*: default color space for byte precision images and files, *texture_paint* if not specified.
- *default_float*: default color space for float precision images and files, *scene_linear* if not specified.

The standard Blender configuration also includes some support for ACES (code and documentation), even though we have a different linear color space. It's possible to load and save EXR files with the Linear ACES color space, and the RRT view transform can be used to view images with their standard display transform. However the ACES gamut is larger than the Rec. 709 gamut, so for best results an ACES specific configuration file should be used. OpenColorIO provides an ACES configuration, though it may need a few more tweaks to be usable in production.

Compatibility

Compatibility with existing files should mostly be preserved. Files that had color management enabled should be entirely compatible, while older files with the color management option disabled are mostly compatible but different for vertex colors and viewport colors.

See Also

- **Developer Documentation**
- User:Sobotka/Color_Management

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Motion Blur

Blender's animations are by default rendered as a sequence of *perfectly still* images. This is unrealistic, since fast-moving objects do appear to be 'moving', that is, blurred by their own motion, both in a movie frame and in a photograph from a 'real-world camera'. To obtain such a Motion Blur effect, Blender can be made to render the current frame and some more frames, in between the real frames, and merge them all together to obtain an image where fast-moving details are 'blurred'.

The Human Eye

Our brains process about 15 images from each eye in parallel each second. My brain cognates those images together and I perceive motion by comparing the two. If something is moving fast enough, I perceive it to be a blur (either because my rods have some latency in reacting to light, or my brain, in overlaying and differencing the images, somehow merges them in a mix sort of fashion). The POINT IS, I *perceive* a motion blur.

In Film

To keep us from seeing jumpy motion pictures, we simply doubled the frame rate to 30 frames per second (fps) (24 fps EU). So, the shutter is basically open for a 30th of a second and the film is exposed to the world for that length of time. As things moved in the real world during that time, the film exposure caused the image of the moving thing to be physically blurred or smeared on that frame. When developed and shown, we physically see an image that is blurred. The POINT IS, I *see* a blurred image.

In CG

In CG, when a frame is rendered, the computer knows exactly where everything should be, and renders it as such. From frame to frame, an object is location A in frame 1, and location B in frame 2. When we show you these two frames at speed (30 fps), the image appears jumpy to us, because somewhere between the eyeballs and the film, there isn't that same blurring as in the real world and film, and we can tell.

Motion Blur in Blender

So, how can we make a blurry CG image? Blender has two ways to achieve Motion Blur:

Sampled Motion Blur

This method is slow, but produces better results. It can be activated in the motion blur section in the render options panel.

Motion Samples
: Set the number of samples to take for each frame.

Shutter
: Time Taken in frames between shutter open and close.

Vector Blur

Vector Blur is faster but sometimes has unwanted side-effects - which can be avoided, though. Vector blur is a process done in compositing, by rendering the scene without any blur, plus a pass that has movement information for each pixel. This information is a vector map which describes a 2d or 3d direction and magnitude. The compositor uses that data to blur each pixel in the given direction.

Examples

To better grasp the concept, let's assume that we have a cube, uniformly moving 1 Blender unit to the right at each frame. This is indeed fast, especially since the cube itself has a side of only 2 Blender units.

Image 1 shows a render of frame 1 without Motion Blur; *Image 2* shows a render of frame 2. The scale beneath the cube helps in appreciating the movement of 1 Blender unit.

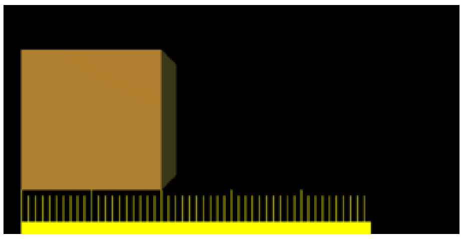


Image 1. Frame 1 of moving cube without Motion Blur.

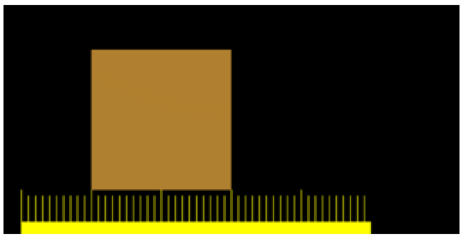


Image 2. Frame 2 of moving cube without Motion Blur.

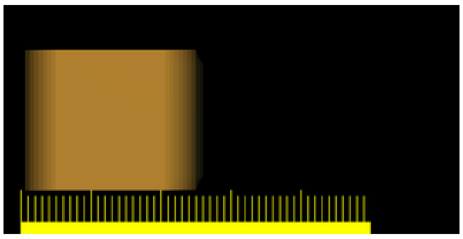


Image 3. Frame 1 of moving cube with Motion Blur, 8 samples, Shutter=0.5.

Image 3 on the other hand shows the rendering of frame 1 when Motion Blur is set and 8 'intermediate' frames are computed. Shutter is set to 0.5; this means that the 8 'intermediate' frames are computed on a 0.5 frame period starting from frame 1. This is very evident since the whole 'blurriness' of the cube occurs half a unit before and half a unit after the main cube body.

Image 4 and *Image 5* show the effect of increasing Bf values. A value greater than 1 implies a very 'slow' camera shutter.

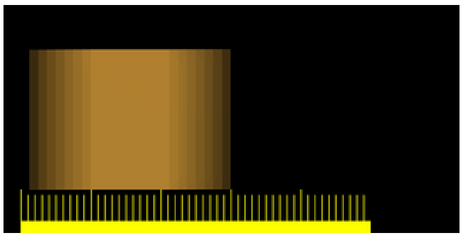


Image 4. Frame 1 of moving cube with Motion Blur, 8 samples, Shutter=1.0.

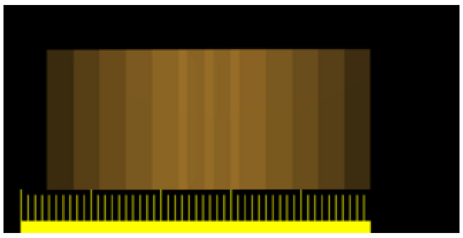


Image 5. Frame 1 of moving cube with Motion Blur, 8 samples, Shutter=3.0.

Better results than those shown can be obtained by setting 11 or 16 samples rather than 8, but, of course, since as many *separate* renders as samples are needed, a Motion Blur render takes that many times more time than a non-Motion Blur one.

Hints

If Motion Blur is active, even if nothing is moving in the scene, Blender actually 'jitters' the camera a little between an 'intermediate' frame and the next. This implies that, even if Anti-Aliasing is off, the resulting images have nice Anti-Aliasing. MBLUR-obtained Anti-Aliasing is comparable to Anti-Aliasing of the same level, but is generally slower.

This is interesting since, for very complex scenes where a level 16 Anti-Aliasing does not give satisfactory results, better results can be obtained using *both* Anti-Aliasing and MBlur. This way you have as many samples per frame as you have 'intermediate' frames, effectively giving oversampling at levels 25, 64, 121, 256 if 5,8,11,16 samples are chosen, respectively.

Optimizing Render Performance

"A watched pot never boils" is the old saying, but you may wonder why your render takes so long to create, or worse, crashes mid-way through! Well, there is lots going on and lots you can do to speed up rendering or enable a complicated render to complete. Also, it is possible to render a very complicated scene on a mediocre PC by being "render-smart". Here's a "top ten" list of things to do or not do in order to speed up rendering or even avoid crashes during scene render. Some options may decrease the quality of your render, but for draft renders you may not care.

If you get the message "Malloc returns nil", in plain English that means the memory allocator tried to get more physical memory for Blender but came back empty-handed. This means that you do not have enough memory available to render the scene, and Blender cannot continue. You will need to do one or more of the following tasks on this page in order to render.

Hardware Improvements

1. Get more RAM up to your PC's (motherboard and operating system) limit. Presently, Blender can use up to 8GG (giga-giga) of physical memory (64-bit address space), but most PCs can only handle 4G of RAM.
2. Upgrade your CPU to a multi-core/multiprocessor
3. Upgrade your OpenGL video drivers
4. Get a faster memory bus
5. Get faster memory, up to your PC's motherboard limit. 667MHz memory is 30% slower than 800MHz.
6. Use or set up a render farm using all available PCs in your house, or use a render farm such as BURP.

Operating System Configuration

1. Increase Blender's processing priority through your OS.
2. Increase your swap file space used by the OS for memory swapping. Also called virtual memory pagefile size, up to the size of your physical memory.
3. Upgrade to a 64-bit operating system (if you're not already using one).
4. Exit or stop any background processes, like virus scanners, BOINC, Real, even "inactive" ones like Quicktime, as they can randomly start up to look for updates. They also take up RAM.
5. Disable network connections to stop random pinging traffic and refresh traffic.
6. Stop listening to Internet radio, and close web browsers, especially any multi-media (audio/video/game) sites.
7. Close down all other running applications, like Word.
8. Exit all TSRs (those icons on your system tray) and widgets, any background processes, and virus scanners.

Blender Settings

1. Increase the MEM Cache Limit in the User Preferences System & OpenGL tab.
2. Upgrade to an optimized Blender build, especially if you have a modern chip that supports SSE2 - render times are **30% faster** using an optimized build.
3. Switch to an Orthographic camera, and render your own "parts" of the scene as separate images, and then paste those parts together in GIMP. An old trick in making your own panorama with a real camera is to take three or so pictures of a very wide (beach sunset) scene, where you take one picture, rotate to the right, snap another, then another, and when you get the pictures developed, you overlap them to make a very wide landscape image. Do the same in Blender: render out one shot to a file, then move the camera to look at a different area of the scene, and render that shot. Each shot will be of a smaller area and thus take in fewer polygons/faces. Be sure that when you position your camera that you snap overlapping shots, so that you can then match them up. If you don't want to use GIMP, you can use compositing nodes and the Translate node to match them up in Blender.
4. Minimize the render window (and Blender if rendering to an internal window). ATI users report dramatic speedup on a per frame basis, which adds up over the frame range.
5. Use the Big Render script to render sub-sections of the overall image, and then paste them together.
6. Make a customized build. For example, comment out calls to check_non_flat_quads in convertblender.c; in some cases can make a noticeable difference of 20-40%.

Scene and Specific Objects

1. Remove lamps, or move them to unrendered layers, or tie them to layers.
2. Turn off some lamp's shadows, using only one or two main sun lamps to cast shadows. A few "shadows only" lights will render faster than every light having shadows on.
3. Use Buffer Shadows rather than ray-traced Shadows
4. Bake your shadows using Render Baking Full Render bake on surfaces that do not move. Use that texture for that mesh, then disable shadows for that material.
5. Simplify meshes (remove polygons). The more vertices you have in camera, the more time it takes to render.
6. Remove Doubles, or use the Decimator mesh edit feature.
7. Remove Subsurf and Multires modifiers.
8. Delete backsides of meshes (removing unseen geometry).
9. Render just a few objects at a time; in the beginning of your project, render the background objects and sets that will not change and will always be in the background.
10. Put the buildings on another layer, and through render layers, don't render them. Then composite them back in later.
11. Make the camera static so that you can better accomplish the above two ideas.
12. Avoid use of Area lights.
13. Make materials Shadeless.
14. Render Bake AO and textures, and then make those materials Shadeless.
15. Decrease the Clip distance for spot lights.
16. Decrease the Clip distance for the camera.
17. Turn off world AO.
18. Turn off Material SSS.

19. Use smaller image textures. A 256x256 image takes only 1% of the memory that a 2k image does, often with no loss of quality in the ultimate render.
20. Reduce Subsurf. Each level quadruples (4x) the number of faces from the previous level.
21. Reduce Multires.
22. Make a matte render of background objects, like buildings, and put the image of them on a billboard in the scene instead of the object themselves. This will reduce vertex/face count.
23. if you have lots of linked instances of an object, use DupliFaces, as these are instanced. If you have 100 of them, Blender will only store the geometry for 1 (Instances themselves take a small amount of memory).

Render Settings

- Output Panel
  1. Disable Edge rendering.
  2. Save Buffers.
  3. Render to an Image Editor window, not a popup. Render Window.
  4. Use multiple Threads on a multi-core CPU (with multiple Parts).
- Render Layers Panel
  1. Render only the Layers of interest.
  2. Render with all lights set to one simple spot (enter its name in the Light: field).
  3. Render with one material override (enter its name in the Mat: field).
  4. Disable unnecessary Render Passes, such as Z, or only render the pass of interest, such as Diffuse.
- Render Panel
  1. Turn off Shadows.
  2. Turn off Environment Mapping.
  3. Turn off Panoramic Rendering.
  4. Turn off Raytracing.
  5. Turn off SSS Subsurface Scattering.
  6. Turn off or lower oversampling/aliasing OSA.
  7. Turn off or lower Motion Blur.
  8. Render in Parts. This will also allow you to render HUGE images on a weak PC. On a multi-core PC, it will assign a thread to each part as well.
  9. Increase the octree resolution.
  10. Render at a percentage size of your final resolution (like 25%).
  11. Turn off Fields rendering.
  12. Use Border rendering to render a subset of the full image.
- Anim Panel
  1. Decrease the frame count of the animation (and use a lower framerate for the same duration of animation). For example, render 30 frames at 10 frames per second for a 3-second animation, instead of 75 frames at 25 frames per second.
- Bake Panel
  1. Bake Full Render - create a UV Texture that colors the objects based on materials, and then use that UV Texture shadeless instead of the material.
  2. Bake Ambient Occlusion only.
  3. Bake textures for objects.
  4. Baking Normals or Displacement does not speed up render time, and are used for other things.
- Format Panel
  1. Render at a lower resolution. Smaller pictures take less time to render.
  2. Choose a faster CODEC or CODEC settings.
  3. Render in black and white (BW button).
  4. If using FFMPEG, do not activate Multiplex audio.
  5. If using FFMPEG, Autosplit Output (Video panel button).
  6. Render only RGB if you just need color; the A channel (RGBA button) takes more memory and is unused when saving a movie file.

Multi-Pass Compositing

Another strategy that can be used to address the problem of long (re-)render times is to structure your workflow from the ground up so that you make aggressive use of *compositing,* as described in the "Post-Production" section. In this approach, you break down each shot into components that can be rendered separately, then you combine those separately-rendered elements to achieve the finished clip. For instance:

- If the camera isn't moving, then neither is the background: only a single frame is needed. (The same is true of any non-moving object within the frame.) These individual elements, having been generated *once,* can be re-used as many times as necessary over as many frames as necessary.
- Both shadows and highlights can be captured separately from the objects that are being illuminated or shadowed, such that the intensity, color, and depth of the effect can be adjusted later without re-rendering.
- Start by using lights that do not cast shadows. (Shadow calculations are big time-killers.) Then, use "shadow-only" lights (which cast shadows, but do not cast light) to create shadows *only* where you judge that they are actually necessary. (It is very often the case that only a few of the shadows which could exist in the scene actually matter, and that the rest of them simply won't be noticed.)
- Tricky lighting situations can be avoided by handling the objects separately, then combining the individually-rendered clips and "tweaking" the result.

This is a very familiar idea. Modern sound recordings, for example, always use a "multi-track" approach. Individual components of the song are captured separately and in isolation, then the components are "mixed" together. The "final mix" then goes through additional processing stages, called *mastering,* to produce the finished product(s). (In fact, the features and design of modern sound-processing software are directly comparable to that of Blender's node-based compositor.)

There are compelling advantages to this approach:

- You have options. If something is "not quite right," you don't necessarily have to start over from scratch.
- In practice, the deadline-killer is *re*-rendering, which ordinarily must be done (in its entirety) just because "'one little thing' about the shot is wrong." Compositing helps to avoid this, because (ideally...) only the specific parts that are found to be in error must be repeated. (Or, maybe, the error can be blocked out with a "garbage matte" and a corrected version can be inserted in its place. No one will ever know!)
- It's also possible that you find yourself saying, "okay, that's *almost* what I wanted, but now I'd like to *add* this and maybe *take away* that." A compositing-based approach enables you to do just that, and furthermore, to do so *non-destructively.* In other words, having generated the "addition" (or the "mask") as a separate channel of information, you can now fine-tune its influence in the overall "mix," or even change your mind and remove it altogether, all without permanently altering anything.
- By and large, these stages work *two*-dimensionally, manipulating what is by that time "a raster bitmap with R, G, B, Alpha *(transparency...)* and Z-Depth information," so they're consistently fast.
- Since each discrete rendering task has been simplified, the computer can carry them out using much fewer resources.
- The tasks can be distributed among several different computers ... even less-powerful ones (like the two older machines that are sitting in your closet right now because you can't get rid of them).
- "After all, the scene doesn't actually have to be *physically perfect,* to be *convincing.*" A compositing-based approach lets you take full advantage of this. You can focus your attention (and Blender's) upon those specific aspects of the scene which will actually make a noticeable difference. It is possible to save a considerable amount of time by consciously choosing to exclude less-important aspects which (although "technically correct") probably won't be noticed.

Of course, this approach is not without its own set of trade-offs. You must devise a workable asset-management system for keeping track of exactly what material you have, where it is, whether it is up-to-date, and exactly how to re-create it. You must understand and use the "library linking" features of Blender to allow you to refer to objects, nodes, materials, textures and scenes in a carefully-organized collection of other files. You need to have a very clear notion, *in advance,* of exactly what the finished shot must consist of and what the task breakdown must be. You must be a scrupulous note-taker and record-keeper. But sometimes this is the best way, if not the *only* way, to accomplish a substantial production.

Distributed Render Farm

There are several levels of CPU allocation that you can use to decrease overall render time by applying more brainpower to the task.

Multi-Threading

First, if you have a multi-core CPU, you can increase the number of threads, and Blender will use that number of CPUs to compute the render.

Frame Ranges

Second, if you have a local area network with available PCs, you can split the work up by frames. For example, if you want to render a 200 frame animation, and have 5 PCs of roughly equal processing power, you can allocate PC#1 to produce frames 1-40, PC#2 to frames 41-80, and so on. If one PC is slower than the others, simply allocate fewer frames to that PC. To do LAN renders, map the folder containing the .blend file (in which you should have packed your external data, like the textures, …) as a shareable drive. Start Blender on each PC and open the .blend file. Change the Start and End frame counts on that PC, but do not save the .blend file. Start Rendering. If you use relative paths for your output pathspec, the rendered frames will be placed on the host PC.

Collaborative Rendering

Third, you can do WAN rendering, which is where you email or fileshare or Verse-share the .blend file (with packed data!) across the Internet, and use anyone's PC to perform some of the rendering. They would in turn email you the finished frames as they are done. If you have reliable friends, this is a way for you to work together.

Remote Renderfarms

Fourth, you can use a render farm service. These services, like BURP, are run by an organization. You email them your file, and then they distribute it out across their PCs for rendering. BURP is mentioned because it is free, and is a service that uses fellow BlenderHead PCs with BOINC-type background processing. Other services are paid subscription or pay-as-you-go services.

Description

Networks in Blender rendering

Goals

- Transparency
- Flexibility

Instructions

Since version 2.6, the rendering system must be enabled in the user preferences, User Preferences-> Addons. .

GUI

Master

On a same machine start a master server.

- Start Blender, selected renderer by network located in the menu drop down from the top of the window of the information (next to "scene").
- (Make sure that you have selected rendering mode) Select master (Master) as operating mode.
- optionally specify the IP address of the interface of listening as well as the port. Let su [default] if you want that the server is listening on all interfaces of the network for the specified port.
- Press Start (it will open a window of empty rendering). The line of the render state will display the server actions.
- The master (Khadijapandu) guide to the stop by pressing ESC, as cancel a normal rendering.

Web interface Master

At startup, the master will also present a web interface that provided more information on slave machines and monitors.There are currently two web interface. The former can be accessed using the URL following http [s] :/ / master_ip_address:master_port. The new is based on the jquerty and development can be accessed the URL following url scheme http [s] :/ / master_ip_address:master_port/html/newui. All information regarding the new interface is located in the following link here.

(Slave(s))

On other machines, made throw slaves

- Start Blender, then to the rendering engine use made by network.
- (Make sure you have selected rendering parameter) Select slave like mode of operation.
- in Option specify the IP address of the master and the port leave [by default] server if you want that slaves automatically detect the master.
- Press on "Start"(Il ouvrira une fenêtre vide de rendu).The render status bar will display the actions the slave.
- The slave is running to stop by pressing ESC. as the cancellation of a normal rendering.

Client

On other machines, made throw slaves

- Start Blender. Confirm your render settings (size, etc.).
- Save the file (it sends the last saved file at this point).
- Select Network Render as Render Engine.
- Select Client as mode of operation.
- Do one of the following:
  - Specify the IP address of the master server as well as the port.
  - Press the Refresh button underneath the address to automatically detect the Master server from its broadcast.
- Press Send Job to dispatch the animation job to the Master server.
- Whenever you want, Render the Animation (Ctrl-F12) to gather the finished frames. Finished frames will "appear" automatically, while it will pause on ongoing frames.
- You can also hit Render on any frame of the animation and it will fetch the result from the Master.
- In the simplest example, you can just press "Animation on network" and wait for the frames to come in. Total render time should be close to inverse proportional to the number of slaves (minus transfer times).

It is possible to run Master, Client and slave on one System.

Command Line

- Configure master as described previously. Instead of clicking "Start Service" save the file (i.e.: master.blend).
- Do the same with the slave setting (i.e.: slave.blend).
- Use background rendering to start the master and slaves like so:
  - *blender -b master.blend --addons netrender -a -noaudio -nojoystick*
  - *blender -b slave.blend --addons netrender -a -noaudio -nojoystick*
- Master and Slaves can be stopped with Ctrl-C (it is recommended to stop the Slaves before the Master).

The option *--addons netrender* ensures that the network render addon is loaded. The options *-noaudio -nojoystick* are optional. They're only there to prevent some warnings.

Extra

Full multilayer render results are used, so the final results should be exactly the same as a local render. You don't have to specify this as the output in the original file; it's done on the slaves automatically.

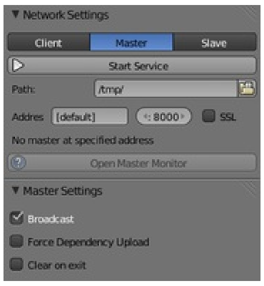
Testers are invited to contact **theeth** via IRC (`#blendercoders`) or by email.

Settings

NetRender as a Render Engine

The Render Engine drop-down is located in the Info window at the top of the Blender window. This is where you select Network Render to access NetRender features.

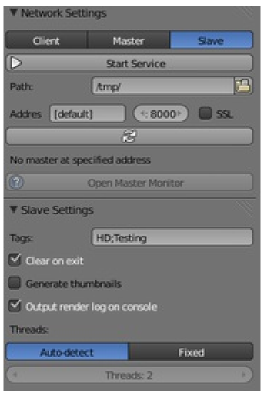
Master


▼ Network Settings
Client | [✓ Master] | Slave
Start Service
Path: /tmp/
Addres [default] 8000 ☐ SSL
No master at specified address
Open Master Monitor
▼ Master Settings
☑ Broadcast
☐ Force Dependency Upload
☐ Clear on exit

Master settings

- Network Settings
    - **Start Service**: Starts the Master Server.
    - **Path**: Where the Master will save job files, results, logs and others. It will create a new directory there of the form *master_<pid>* where *<pid>* is the process ID of the Master server. In the root of the folder, a file named "blender_master.data" will be saved to enable resuming a master later.
    - **Server Address**: Address of the network interface that the Master will listen on. *[default]* means listen on all available network interfaces.
    - **Port**: Port that the Master will listen on.
    - **SSL**: Use SSL (https) for connections with slaves and clients. When that option is enabled, two new fields become visible to specify the SSL certificate and key. You can use a self-signed certificate or a certificate provided by a third party like comodo, or verisign. In that case if there is a chain of trust you can put it in the same file as the certificate but the certificate must be put first. The certificate, the chain of trust and key must be provided as PEM files.
    - **Open Master Monitor**: Open a browser to the Web-based Master monitor. Enabled when the Master is running.
- Master Settings
    - **Broadcast**: Broadcast the Master's Address and Port on its local network (every 10s).
    - **Force Dependency Upload**: Forces clients to upload dependency files to the master, instead of using existing local files even if they match client files.
    - **Clear on exit**: Remove the directory created in *Path* when the Master is stopped. Turning on this option prevents resuming a master later if the process is stopped for any reason.

Slave

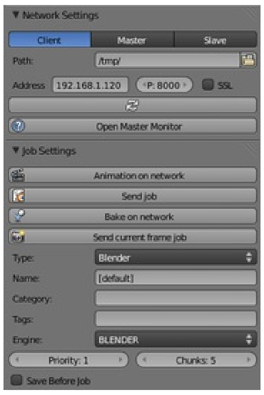

▼ Network Settings
Client | Master | [✓ Slave]
Start Service
Path: /tmp/
Addres [default] 8000 ☐ SSL
No master at specified address
Open Master Monitor
▼ Slave Settings
Tags: HD;Testing
☑ Clear on exit
☐ Generate thumbnails
☑ Output render log on console
Threads:
[✓ Auto-detect] | Fixed
Threads: 2

Slave settings

- Network Settings
    - **Start Service**: Start the Slave node.
    - **Path**: Where the Slave will save job files, results and logs. It will create a new directory there of the form *slave_<id>* where *<id>* is the Slave ID assigned by the Master server.
    - **Server Address**: Address on which the Master listens.
    - **Port**: Port on which the Master listens

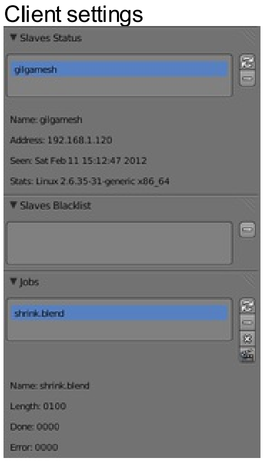
    - **Refresh**: Listen to the Master's broadcast to determine its Address and Port (can take up to 20s).
    - **Open Master Monitor**: Open a browser to the Web-based Master monitor. Enabled when the Master's address is valid.
- Slave Settings
    - **Tags**: Semi-colon separated list of tags assigned to the slave. A slave will only be assigned a job if it has at least all of that job's tags.
    - **Clear on exit**: Remove the directory created in *Path* when the Slave is stopped.
    - **Generate thumbnails**: Create thumbnails of the render result on the Slave (they are otherwise created on demand by the Master).
    - **Output render log on console**: Also output logs from the rendering subprocess to the standard output and not just to render log sent to the master.
    - **Threads**: How many threads the Slave should use for rendering.

Client



Path: /tmp/
Address: 192.168.1.120 P: 8000 SSL
Type: Blender
Name: [default]
Category:
Tags:
Engine: BLENDER
Priority: 1 Chunks: 5

Client settings



Name: gilgamesh
Address: 192.168.1.120
Seen: Sat Feb 11 15:12:47 2012
Stats: Linux 2.6.35-31-generic x86_64

Name: shrink.blend
Length: 0100
Done: 0000
Error: 0000

Slaves and Jobs lists

- Network Settings
    - **Path**: Where the Client will save its temporary render result file.
    - **Server Address**: Address on which the Master listens.
    - **Port**: Port on which the Master listens.
    - **SSL**: Use SSL (https) to communicate with the Master.
    - **Refresh**: Listen to the Master's broadcast to determine its Address and Port (can take up to 20s).
    - **Open Master Monitor**: Open a browser to the Web-based Master monitor. Enabled when the Master's address is valid.
- Job Settings
    - **Animation on network**: Sends the current file as a job to the Master and waits for results (other than the rendering taking place elsewhere, this works like a normal Render Animation).
    - **Send job**: Sends the current file as a job to the Master. The returned job ID becomes the *current job ID*.
    - **Bake on network**: Sends a baking job with all modifiers using a point cache or particle systems in the scene,
    - **Send current frame job**: Sends the current file as a job to the Master with the current frame to be rendered only. The returned job ID becomes the *current job ID*.
    - **Name**: Name of the job. *[default]* uses the name of the blend file.
    - **Category**: Category of the job, *Optional*. Jobs on the Master are also balanced by Categories.
    - **Tags**: Semi-colon separated list of tags assigned to the job. A job will only be assigned to a slave if its tag list contains all of the job's own tags.
    - **Engine**: Render engine to use for rendering this job.
    - **Priority**: Priority of the job. The Priority level is a multiplier that makes the Master count the job as if it were X jobs (i.e.: balancing between a priority 1 and a priority 2 job will make them take 33% and 66% of the workload respectively).
    - **Chunks**: How many frames are dispatched to a Slave as part of a chunk of a job.
    - **Save Before Job**: Forces the current file to be saved to disk before being dispatched as a job.
- Slaves Status
    - **List**: List of all Slaves connected to the Master.
    - **Refresh**: Refresh the Slaves information from the Master
    - **Remove**: Move the selected Slave to the Blacklist.
- Slaves Blacklist

  - **List**: List of all Blacklisted Slaves.
  - **Remove**: Remove the selected Slave from the Blacklist.
- Jobs
  - **List**: List of all jobs on the Master.
  - **Refresh**: Refresh the jobs information from the Master.
  - **Remove**: Remove a job from the Master.
  - **Remove All**: Remove all jobs from the Master.
  - **Get Results**: Get all available frames from the selected job. Results are downloaded as multilayer EXR into the current output directory.

Physics Baking Jobs

Physics baking is a recently added feature in Netrender. It supports dispatching baking jobs for each point cache used in a scene (on a modifier or particle system).

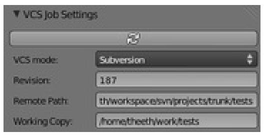
Each point cache is baked individually on a slave; bake ordering and dependencies are not currently supported.

Results can only be downloaded as a zip file from the job's page on the web interface. You then have to unzip it and put the results in the blendcache folder associated with your file and turn on disk cache for modifiers and particle systems that you baked (this step should be done automatically at some point).

The text outputted when baking a point cache is not terribly well-suited for being piped to a log and not very informative, so you won't get a whole lot of information from the job's log file. Changing this would require some change to the baking code directly.

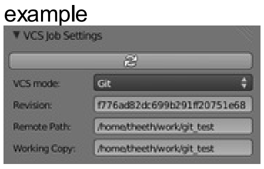
Baking other type of physics (like fluids) should eventually be supported.

Version Control Jobs



VCS mode: Subversion
Revision: 187
Remote Path: th/workspace/svn/projects/trunk/tests
Working Copy: /home/theeth/work/tests

Subversion settings example



VCS mode: Git
Revision: f776ad82dc699b291ff20751e68
Remote Path: /home/theeth/work/git_test
Working Copy: /home/theeth/work/git_test

Git settings example

Using VCS (version control system) as a job type enables you to bypass the usual dependency system used by netrender and rely on a versioning system instead. For more organized productions, this is usually a good idea as it minimizes dependency errors, disk space used and job dispatch time.

Currently, the only two version control systems supported are Subversion (svn) and Git. Adding new ones is relatively easy and will be done when requested.

After selecting a VCS, you have to specify three system-specific settings:

-
  - **Revision**: string used to identify a specific version. (svn: revision, git: commit hash).
  - **Remote path**: remote path where the files can be downloaded from (svn: server url, git: remote repository path from which the slaves can checkout). All job files must be in that folder or one of its subfolders.
  - **Working copy**: working copy root folder. Where the remote files will be downloaded. This is kept between jobs to prevent download of the same file more than once and will only change when jobs require a new revision of specific files from the version control system.

The Refresh button will try to guess those settings to the best of its knowledge.

Notes and Known Bugs

- No shared network space is required between nodes.
- You can dispatch many different files; all results can be retrieved independently. (Save the file after the dispatch if you want to close it and retrieve later.)
- There is very little network error management, so if you close the master first, stuff will break. Same if you enter an invalid address.
- Issue with many dependencies with the same file name: https://projects.blender.org/tracker/index.php?func=detail&aid=25783&group_id=9&atid=498

Yes, I *know*the current workflow is far from being ideal, especially from a professional render farm point of view. I expect Matt to whip me and suggest better stuff. Optimally, I'd like if users could just press "Anim on network", it would automatically dispatch to the network and wait for results, like a local render. All "pro" features should be optional.

Load Balancing

Primary balancing is performed by calculating usage of the cluster every 10s for each job, averaged over time. The next job dispatched is the one with lowest usage (the one that is using the least number of slaves). The priority of a job acts as a divisor, so a job of priority 2 would use a percentage of the cluster as if it were 2 jobs and not just one (i.e.: a job of priority 1 and one of priority 2 sharing slaves will use respectively 33% and 66% of the processing power). On top of that, there's a set of exceptions and first priority rules:

Exceptions

- A single job cannot use more than N% of total slaves, unless it's the only job. That prevents a slow job from starving faster ones. This is set at 75% for now, but should be customizable.

First Priorities (criteria)

- Less than N frames dispatched (prioritize new jobs). The goal of this is to catch errors early.
- More than N minutes list last dispatch. To prevent high-priority jobs from starving others.

To do

- Send job from memory
- Don't depend on render engine choice for visibility
- "Expert" render manager
- Better defined communication protocol
- The option to calculate simulations (cloth, smoke, ...) on a node which would then send point cache to server for dispatch to render
- Pack textures on upload
- Dispatch single frame as tiles

Technical Details

Out of date, read the code and put info here.

Feature List

- support paths instead of files
- client-server-slave: restrict job to specific nodes
- client-server-slave: view node machine stats
- client-server-slave: reporting error logs back to manager (all `stdout` and `stderr` from nodes)
- Cancel jobs
- Restart error frame
- Disable crash report on windows
- Dispatch more than one frame at once (a sequence of frames)
- Blacklist slave that errors on frame after reset
- Multiple paths on job announce
- Delay job until all files accounted for
- Frame range restrictions (ie: send point cache files only when needed for the range of frames)
- Send partial logs to master
- TODO: Set slaves to copy results on network path
- TODO: client-master: archive job (copy source files and results)
- TODO: master-slave: restrict jobs based on specs of slaves.

API Feature Wishlist

This is a list of blender code I would need to make netrender better. Some of them are bugs, some are features that should (hopefully) eventually be there.

- API access to jobs, to be able to run masters and slaves in the background as well as render job notifiers on the client.
- Render result from multilayer image in memory
- Render and load tiles in render results

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Subpages

- Freestyle
- Luxrender
- Yafray

Description

Yafaray, as the lengthened version of its name (Yet Another Free RAYtracer) suggests, is a free, XML speaking, cross-platform raytracer developed by the Yafray team.. It works with many 3D modelling applications (with Wings and Aztec serving as examples), but the focus of this document shall fall upon its use with Blender.

Yafaray is currently available (under the LGPL license) for Windows, Linux (via source code compilation, or .deb or .rpm installation), Mac OSX, and Mac Intel; and installation packages, as well as Yafray's source code, can be downloaded here.

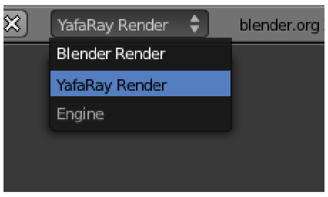
Options

Blender releases between 2.34 and 2.5x have the option to call Yafray in place of Blender's internal renderer (assuming it's installed). This can be done by selecting "Yafaray Render" from the drop-down engine selector menu on the Info header.

Render Pipeline
When Yafaray is used, it is inserted into the pipeline before any compositor or sequencer actions, because it is the renderer, and the compositor and sequencer work on rendered images. The image data given to Yafray is the scene objects, materials, lights, etc. Yafray does not know nor care about render layers and cannot feed Blender's node compositor or sequencer effects, since it takes a completely different approach and cannot produce the different render layers that the Blender internal renderer can. Yafaray renders frames based on Blender scene data.

To use Yafaray with Blender's compositor, render the image using Yafray, and then use the image input node to get that image into the compositor where it can be post-pro. You can then feed that to the Sequencer via the Scene strip and Do Composite enabled. To feed the Sequencer directly from Yafray's output, use the Image strip after Yafaray has completed the render.

Two other panels should appear once Yafaray is selected from this menu, which serve to supply a number of Yafray's options to you (other options are available exclusively as XML code).



Enabling Yafray

XML
: This button, if pressed, will export your scene to a .xml file in your system's 'tmp' directory before Yafray renders it. Useful if you wish to make modifications to the XML file, or render the scene from a command line interface. Should you wish, however, to view Yafray's progress during a render in Blender's render window, it's better to unclick this button.

AutoAA
: This option allows you to toggle between manual and automatic control of anti-aliasing options in the scene. Anti-aliasing is similar to Blender internals OSA, which in effect dictates the accuracy of the edges in the render.

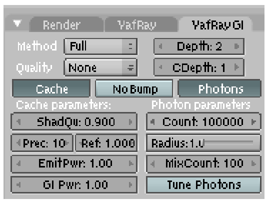
  In cases where you may need to manually control the anti-aliasing options (which becomes necessary if you wish to make use of Yafray's depth-of-field option), it's useful to remember that increasing the amount of samples per pass will increase the accuracy of the edges in the final render; decreasing the amount of samples per pass will, as you'd expect, decrease the accuracy, causing edges in the scene to seem rough and jagged.

Proc.
: This option allows you to select the number of processors Yafray is allowed to make use of. For those of us who aren't lucky enough to have multiple processors, it's best to leave this option as its default.

Gam.
: This option allows for manual correction of gamma values in the scene. The default (1) turns this option off.

Exp.
: This option allows for manual adjustment of exposure levels in the scene. A more in-depth explanation of this option will come later.

Global Illumination



Method Full
Quality None
Depth: 2
CDepth: 1
ShadQu: 0.900
Prec: 10
Ref: 1.000
EmitPwr: 1.00
GI Pwr: 1.00
Count: 100000
Radius: 1.0
MixCount: 100

Global illumination settings

The next tab, titled "Yafray GI", provides a selection of methods with which Yafray is able to light a scene. Methods available are:

Full
: This method works well for most scenes, most notably indoor scenes, where the use of photons becomes appropriate.

Skydome
: This method is more suited to outdoor scenes.

Cache
: Clicking the **cache** button speeds up rendering by allowing Yafray to be more selective in its distribution of samples. When this button's depressed, Yafray renders a pre-pass to determine the most suitable allocation of samples, before rendering the image itself, increasing the efficiency of the render.

The cache button then reveals three more options.

ShadQu
: This option allows for greater control over the quality of shadows. By increasing this option from its default (0.900), you also increase the number of samples taken in shadowed areas, which in turn not only increases the quality of shadows in the scene, but also increases render times.

Prec
: This option sets the maximum number of pixels per square without samples. By decreasing this option from its default (10), you increase the number of samples taken in the scene. Decreasing this option also increases render times.

Ref
: This option allows the user to specify the threshold to refine shadows. By decreasing this option from its default (1.000), you invite Yafray to increase the number of passes taken to distribute samples in shadowed areas, thereby increasing the quality of the shadows in the scene, and increasing render times.

Examples

Starting with the default Blender setup, enable Yafray in the Rendering Buttons panel (F10), deselect the XML option in the "yafray" tab, and select the "full" method from the "yafray GI" tab, and set the quality to "low". Then click "Render" (F12).

Console output

Provided the environment allows it, Yafray should output information to the console window (in Windows, Blender opens alongside a console window by default. In GNU/Linux, however, to view the console output, you'll need to start Blender from the console, usually by typing "blender" into a terminal emulator window).

If you switch to the console after the render is completed, you should (provided the "cache" option's enabled) notice something similar to this:

Console output

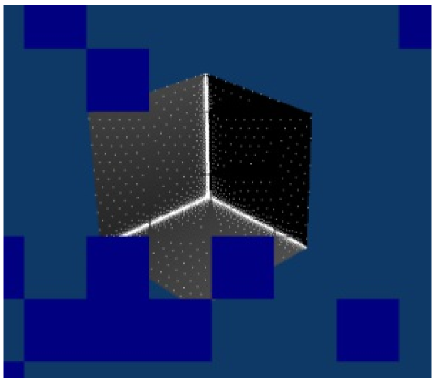
```
Launching 1 threads

Fake pass: [#############]
534 samples taken

Render pass: [#############]

render finished
```

Output description

The render is split up into two separate passes. The first "fake" pass is made as a direct result of the "cache" option being enabled, and its purpose is to determine the best distribution of samples in the scene (without the cache option enabled, the samples are distributed evenly within the scene). The number of samples is then output onto the next line.

The next pass is the "real" render pass, where Yafray renders the image based on the sample map created in the previous pass.

Render window output

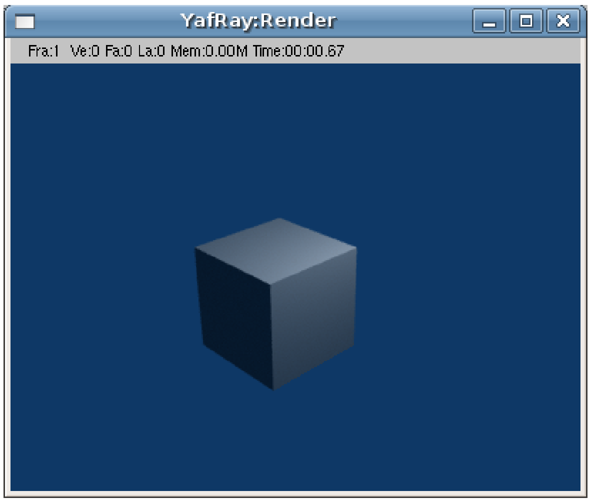

Greater samples in shadowed areas

Now we'll look at Yafray's output to the render window during the render.

Provided the XML option is turned off, Yafray will continually update its visual output to the render window, much like Blender does. The image to the right was captured during the "fake" pass stage of the render, and the white dots represent the allocation of samples in the scene. Notice how the samples are only placed in areas of the scene that are directly affected by light, meaning that, in the demonstration image, only the parts of the scene with a surface are considered.

This also means that in shadowed areas of the scene, the number of samples is greater.

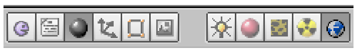
You can notice that the density of white dots which, as I pointed out earlier, represent the number of samples per pixel in that area of the image, is greater in areas that are likely to be shadowed (in this case, I deleted the vertex of the cube closest to the camera, revealing inside edges, which aren't as exposed to the light).



Basic Yafray render

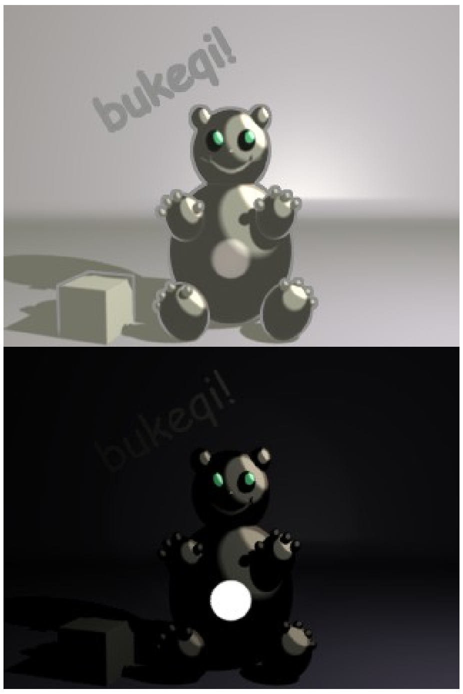
The rendered image

You'll notice how the cube, despite Blender's default gray material being applied, has been colored blue. This is because the Full method is affected by the "world" color of the scene, which, again as Blender's default, is blue. To change this, switch to the "shading" panel (F5), and select the little world icon. To have materials show properly, set the world shader to white.


Selecting the world shader

Notes

Amount of Light



YafRay deals with light completely differently than the Blender Internal Renderer, and apparently light intensity needs to be pumped by large amounts for YafRay. The images reflect a Blender Internal render, a Yafray render without Global Illumination (GI), and one with Full GI. As you can see, results vary widely based on the illumination method chosen.

A solution is to use very large Area lamps (Square, 100 Size but Samples at only 4, Energy 10) for softer shadows, in combination with a Sun lamp at much lower Energy value (less than 1.0) if you want a distinct shadow edge. Sun lamps seem to provide much greater intensity than Area lamps in YafRay but the shadow edges are quite harsh.

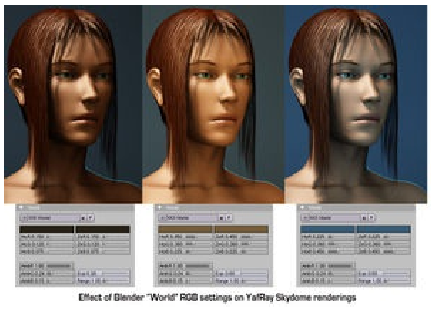
Try using the Skydome setting for the YafRay GI because with Full GI you may get weird blotchy artifacts that no one seems to know how to remedy, but may be related to the scale of my Blender scene, which is 1BU = 1cm, with a figure built to life-size. You'll be doing something like this as well if you build a scale model to match camera perspectives.

Blender World parameters may include a small AO setting which YafRay does seem to take into account, so you might try adding some in your scene. Also be aware that the World Sky colors (Ho & Ze) are treated as a "hemi" light source, and will color your scene accordingly when using Skydome -- play with these RGB values to perhaps boost the overall lighting intensity by "filling in" with GI. In the pics below, the World lighting settings were doubled for the render on the right.

Everything seems to need to be boosted for YafRay -- some Materials look very dull unless you "double up" some of the components (such as by using an image texture twice with "Add"), and the RGB & Shader tab settings are very different from what you would use with the Internal renderer.

You can also adjust the EmitPwr and Exp settings in the YafRay renderer tabs to compensate for the lighting differences. It gets to be quite a juggling act. The plus side is that you are able to get lighting of a much richer character for a scene, so it can be worth the trouble.

SkyDome


Effect of Blender "World" RGB settings on YafRay Skydome renderings

Various coloring effects based on World settings

Using the Blender Internal (BI) renderer, the only way to get the world Horizon, Zenith, or Textured color to affect the material color is to use Ambient Occlusion set to Sky Color or Sky Texture; otherwise (without AO) it only affects the color of the background. The only variable to directly affect the final object coloration in Blender Internal is the color of Ambient light, and then each material can receive a specified amount of that ambient light (by default 50%). The color of the ambient light in BI cannot be varied over the height of the image and is applied uniformly to the subject. Ambient Occlusion, based on the settings, affects the color of the model based on its geometry.

In Yafray, however, a key difference is that the color of all of these matter, as shown in the example. The example has the same material (the skin and hair) rendered using different **Horizon and Zenith** colors. Each of these, in effect, change the ambient light cast onto the subject. If the Zenith was darker, as is usually the reality, the tops of the model would be darker than the the lower portions. Using the color of the sky and horizon to affect the lighting of subjects lends a much more realistic blending of a subject into the environment, leading to more photorealistic results.

To achieve the same effect in Blender, you can use Ambient Occlusion, or light your subject with Hemisphere lamps which are the same color as your sky zenith and horizon.

Page status (reviewing guidelines)

Void page
Proposed fixes: none

Cycles Render Engine

Cycles is a ray tracing renderer focused on interactivity and ease of use, while still supporting many production features. Developer documentation is also available.

Getting Started

Cycles is bundled as an addon that is enabled by default. To use Cycles, it must be set as the active render engine in the top header. Once that is done, interactive rendering can be started by setting a 3D view editor to draw mode Rendered. The render will keep updating as material and object modifications are done.

Reference

Freestyle

Freestyle is a render engine for creating Non-Photorealistic Line Drawing images from 3D scenes. It is designed as a programmable interface to allow maximum control over the style of the final drawing: the user "programs" how the silhouettes and other feature lines from the 3D model should be turned into stylized strokes using a set of programmable operators dedicated to style description.

Using Freestyle in Blender

See Here for documentation on using Freestyle in Blender.

Cycles Render Engine

Cycles is a ray tracing renderer focused on interactivity and ease of use, while still supporting many production features. Developer documentation is also available.

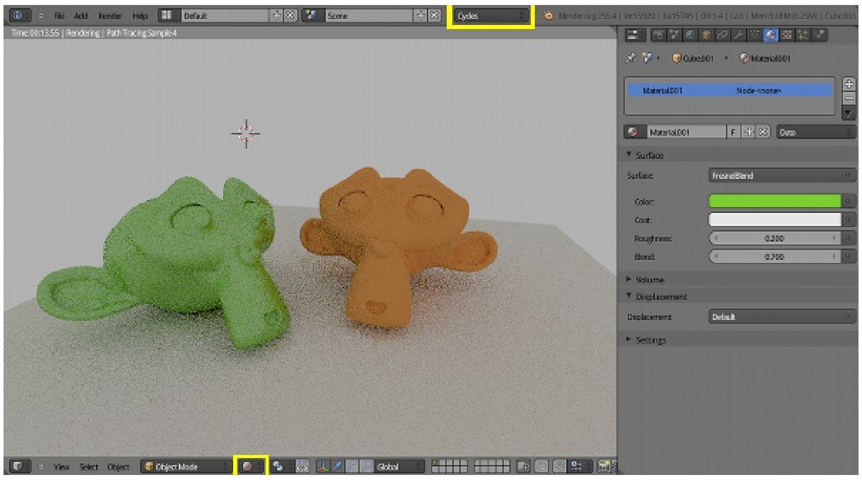
Getting Started

Cycles is bundled as an addon that is enabled by default. To use Cycles, it must be set as the active render engine in the top header. Once that is done, interactive rendering can be started by setting a 3D view editor to draw mode Rendered. The render will keep updating as material and object modifications are done.

Reference

Getting Started

Cycles is bundled as an addon that is enabled by default. To use Cycles, it must be set as the active render engine in the top header. Once that is done, interactive rendering can be started by setting a 3D view editor to draw mode Rendered. The render will keep updating as material and object modifications are done.



GPU Rendering

To see if and how you can use your GPU for rendering, see the documentation on GPU Rendering.

Tutorials

Here you can find some tutorials for Cycles.

Tutorials

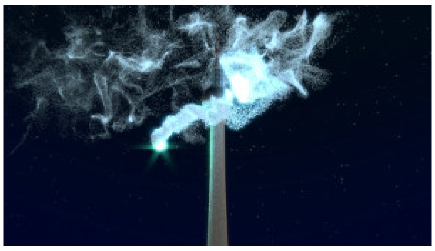
- Cycles Complete Overview
- Cycles Basics Tutorial Series

- Introduction to Cycles (Blender Guru)

- Introduction to the Cycles Render Engine (Blender Cookie)

- Volumetric Absorption in Cycles

- Advanced Particle Trail with Cycles



- An Animated Pin Toy (BlenderDiplom)

- How to create hoar frost in Cycles (BlenderDiplom)

- Create a Realistic Water Simulation (Blender Guru)

- Advanced Organic Material Setup with Maps (BlenderDiplom)

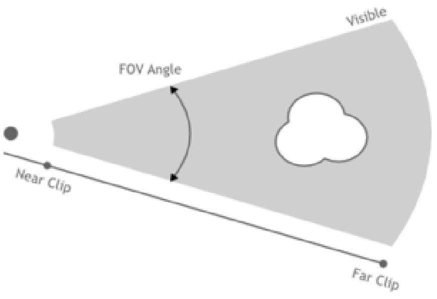
- Rendering a Scene of Wooden Barrels (Blender Cookie)

- Rendering Nuts and Bolts with Cycles (Free 3D Tutorials)

Camera

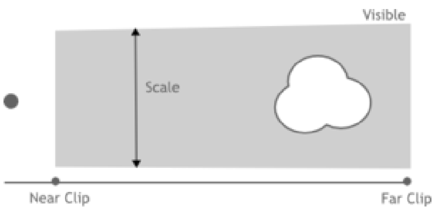
Perspective

Lens Size and Angle
: Control the field of view angle.



Orthographic

Scale
: Controls the size of objects projected on the image.



Panoramic

Cycles supports Equirectangular and Fisheye panoramic cameras. Note that these can't be displayed with OpenGL rendering in the viewport; they will only work for rendering.

Equirectangular

Render a panoramic view of the scenes from the camera location and use an equirectangular projection, always rendering the full 360° over the X-axis and 180° over the Y-axis.

This projection is compatible with the environment texture as used for world shaders, so it can be used to render an environment map. To match the default mapping, set the camera object rotation to (90, 0, -90) or pointing along the positive X-axis. This corresponds to looking at the center of the image using the default environment texture mapping.

Fisheye

Fisheye lenses are typically wide angle lenses with strong distortion, useful for creating panoramic images for e.g. dome projection, or as an artistic effect. The Fisheye Equisolid lens will best match real cameras. It provides a lens focal length and field of view angle, and will also take the sensor dimensions into account.

The Fisheye Equidistant lens does not correspond to any real lens model; it will give a circular fisheye that doesn't take any sensor information into account but rather uses the whole sensor. This is a good lens for full dome projection.

Lens
: Lens focal length in millimeter.

Field of View
: Field of view angle, going to 360 and more to capture the whole environment.

Depth of Field

Aperture Type
: Method with which to specify the size of the camera opening through which light enters. With Radius the radius of the opening can be specified, while F/Stop specifies the size relative to the camera focal length, a measure more common in photography. Their relation is: *aperture radius = focal length / (2 f-stop)*

Aperture Size
: Also called lens radius. If this is zero, all objects will appear in focus, while larger values will make objects farther than the focal distance appear out of focus.

Aperture F/Stop
: Also called F-number or relative aperture. Lower numbers give more depth of field; higher numbers give a sharper image.

Aperture Blades
: If this setting is 3 or more, a polygonal-shaped aperture will be used instead of a circle, which will affect the shape of out of focus highlights in the rendered image.

Aperture Rotation
: Rotation of the Aperture Blades.

Aperture Ratio
: Distortion to simulate anamorphic lens. The Aperture Ratio is the height by the width of the bokeh and allows for non-circular bokehs.

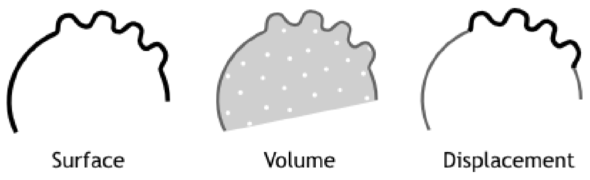
Focal Distance
: Distance at which objects are in perfect focus. Alternatively, an object can be specified whose distance from the camera will be used.

Clipping

Clip Start and End
: The interval in which objects are directly visible. Any objects outside this range still influence the image indirectly, as further light bounces are not clipped. For OpenGL rendering, setting clipping distances to limited values is important to ensure sufficient rasterization precision. Ray tracing does not suffer from this issue much, and as such more extreme values can safely be set.

Materials

Materials define the appearance of meshes, curves and other objects. They consist of three shaders, defining the appearance of the surface of the mesh, the volume inside the mesh, and displacement of the surface of the mesh.



Surface Shader

The surface shader defines the light interaction at the surface of the mesh. One or more BSDFs specify if incoming light is reflected back, refracted into the mesh, or absorbed.

Emission defines how light is emitted from the surface, allowing any surface to become a light source.

Volume Shader

When the surface shader does not reflect or absorb light, it enters into the volume. If no volume shader is specified, it will pass straight through to the other side of the mesh.

If it is defined, a volume shader describes the light interaction as it passes through the volume of the mesh. Light may be scattered, absorbed, or emitted at any point in the volume.

A material may have both a surface and a volume shader, or only one of either. Using both may be useful for materials such as glass, water or ice, where you want some of the light to be absorbed as it passes through the surface, combined with e.g. a glass or glossy shader at the surface.

Displacement

The shape of the surface and the volume inside it may be altered by displacement shaders. This way, textures can then be used to make the mesh surface more detailed.

Depending on the settings, the displacement may be virtual, only modifying the surface normals to give the impression of displacement, which is known as bump mapping, or a combination of real and virtual displacement.

Energy Conservation

The material system is built with physics-based rendering in mind, cleanly separating how a material looks and which rendering algorithm is used to render it. This makes it easier to achieve realistic results and balanced lighting, though there are a few things to keep in mind.

In order for materials to work well with global illumination, they should be, speaking in terms of physics, energy conserving. That means they cannot reflect more light than comes in. This property is not strictly enforced, but if colors are in the range 0.0 to 1.0, and BSDFs are only mixed together with the Mix Shader node, this will automatically be true.

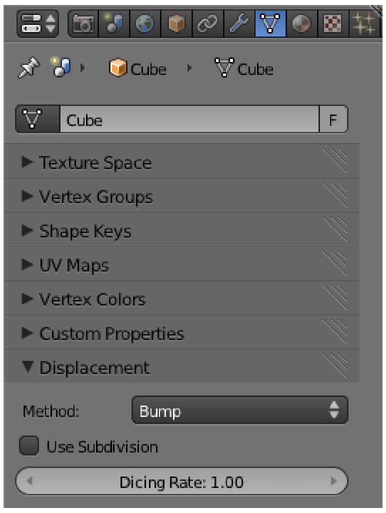
It is however possible to break this, with color values higher than 1.0 or using the Add Shader node, but one must be careful when doing this to keep materials behaving predictably under various lighting conditions. It can result in a reflection adding light into the system at each bounce, turning a BSDF into a kind of emitter.

Displacement

Implementation not finished yet, marked as experimental feature.

The shape of the surface and the volume inside its mesh may be altered by the displacement shaders. This way, textures can then be used to make the mesh surface more detailed.

Type

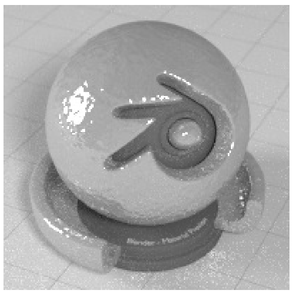


Method: Bump
Use Subdivision
Dicing Rate: 1.00

Depending on the settings, the displacement may be virtual, only modifying the surface normals to give the impression of displacement, known as bump mapping, or a combination of real and virtual displacement. The displacement type options are:

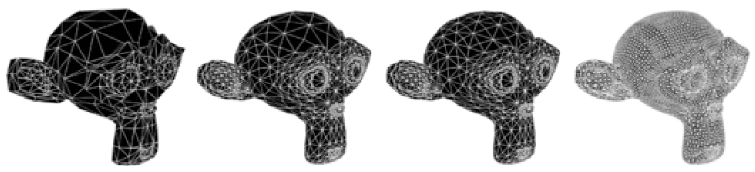
- **True Displacement**: Mesh vertices will be displaced before rendering, modifying the actual mesh. This gives the best quality results, if the mesh is finely subdivided. As a result this method is also the most memory intensive.
- **Bump Mapping**: When executing the surface shader, a modified surface normal is used instead of the true normal. This is a quick alternative to true displacement, but only an approximation. Surface silhouettes will not be accurate and there will be no self-shadowing of the displacement.
- **Displacement + Bump**: Both methods can be combined, to do displacement on a coarser mesh, and use bump mapping for the final details.

Subdivision


Bump Mapped
Displacement

Implementation not finished yet, marked as experimental feature.

When using *True Displacement* or *Displacement + Bump* and enabling *Use Subdivision* you can reduce the **Dicing Rate** to subdivide the mesh. This only affects the render and does not show in the viewport (but does show in *Rendered Shading Mode*). Displacement can also be done manually by use of the Displacement Modifier.


Subdivision Off - On, Dicing Rate 1.0 - 0.3 - 0.05 (Monkeys look identical in viewport, no modifiers)

Surface

The surface shader defines the light interaction at the surface of the mesh. One or more BSDFs specify if incoming light is reflected back, refracted into the mesh, or absorbed.

Emission defines how light is emitted from the surface, allowing any surface to become a light source.

Terminology

- **BSDF** stands for bidirectional scattering distribution function. It defines how light is reflected and refracted at a surface.
- **Reflection** BSDFs reflect an incoming ray on the same side of the surface.
- **Transmission** BSDFs transmit an incoming ray through the surface, leaving on the other side.
- **Refraction** BSDFs are a type of **Transmission**, transmitting an incoming ray and changing its direction as it exits on the other side of the surface.

BSDF Parameters

A major difference from non-physically based renderers is that direct light reflection from lamps and indirect light reflection of other surfaces are not decoupled, but rather handled using a single BSDF. This limits the possibilities a bit, but we believe overall it is helpful in creating consistent-looking renders with fewer parameters to tune.

For glossy BSDFs, **roughness** parameters control the sharpness of the reflection, from 0.0 (perfectly sharp) to 1.0 (very soft). Compared to **hardness** or **exponent** parameters, it has the advantage of being in the range 0.0..1.0, and as a result gives more linear control and is more easily textureable. The relation is roughly: *roughness = 1 - 1/hardness*

Volume

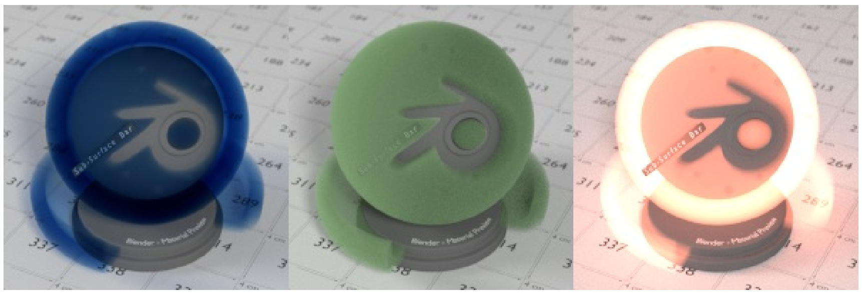
Volume rendering can be used to render effects like fire, smoke, mist, absorption in glass, and many other effects that can't be represented by surface meshes alone.

To set up a volume, you create a mesh that defines the bounds within which the volume exists. In the material you typically remove the surface nodes and instead connect volume nodes to define the shading inside the volume. For effects such as absorption in glass you can use both a surface and volume shader. The world can also use a volume shader to create effects such as mist.

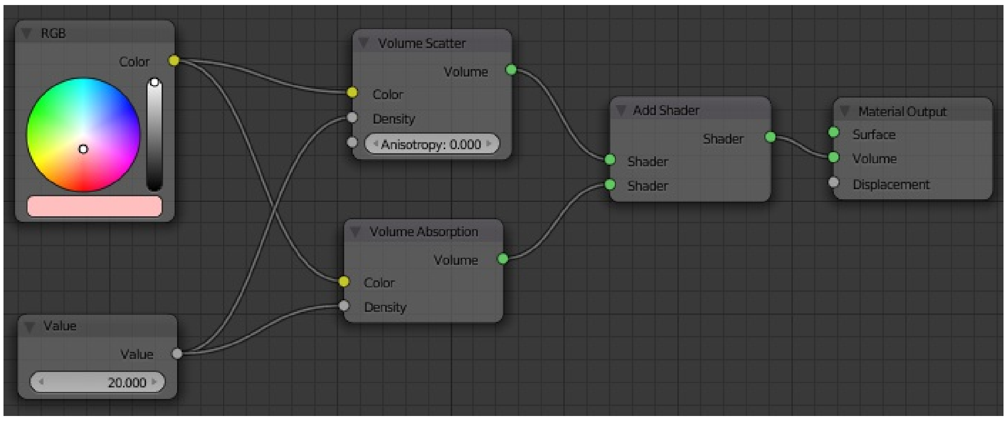
Volume Shaders

We support three volume shader nodes, that model particular effects as light passes through the volume and interacts with it.

- Volume Absorption will absorb part of the light as it passes through the volume. This can be used to shade for example black smoke or colored glass objects, or mixed with the volume scatter node. This node is somewhat similar to the transparent BSDF node, it blocks part of the light and lets other light pass straight through.
- Volume Scatter lets light scatter in other directions as it hits particles in the volume. The anisotropy defines in which direction the light is more likely to scatter. A value of 0 will let light scatter evenly in all directions (somewhat similar to the diffuse BSDF node), negative values let light scatter mostly backwards, and positive values let light scatter mostly forward. This can be used to shade white smoke or clouds for example.
- Emission will emit light from the volume. This can be used to shade fire for example.


Volume Shader: Absorption / Absorption + Scatter / Emission

The following is a basic starting point for a volume shader:



Basic initial setup with Absorption + Scatter volume shaders

The result of above's shader setup

Density

All volume shaders have a density input. The density defines how much of the light will interact with the volume, getting absorbed or scattered, and how much will pass straight through. For effects such as smoke you would specify a density field to indicate where in the volume there is smoke and how much (density bigger than 0), and where there is no smoke (density equals 0).

Volumes in real life consist of particles, a higher density means there are more particles per unit volume. More particles means there is a higher chance for light to collide with a particle and get absorbed or scattered, rather than passing straight through.

Volume Material

Interaction with the Surface Shader

A material may have both a surface and a volume shader, or only one of either. Using both may be useful for materials such as glass, water or ice, where you want some of the light to be absorbed as it passes through the surface, combined with e.g. a glass or glossy shader at the surface.

When the surface shader does not reflect or absorb light, it enters into the volume. If no volume shader is specified, it will pass straight through to the other side of the mesh. If it is defined, a volume shader describes the light interaction as it passes through the volume of the mesh. Light may be scattered, absorbed, or emitted at any point in the volume.

Mesh Topology

Meshes used for volume render should be closed and manifold. That means that there should be no holes in the mesh. Each edge must be connected to exactly 2 faces such that there are no holes or T-shaped faces where 3 or more faces are connected to an edge.

Normals must point outside for correct results. The normals are used to determine if a ray enters or exits a volume, and if they point in a wrong direction, or there is a hole in the mesh, then the renderer is unable to decide what is the inside or outside of the volume.

These rules are the same as for rendering glass refraction correctly.

Volume World

A volume shader can also be applied to the entirely world, filling the entire space.

Currently this is most useful for night time or other dark scenes, as the world surface shader or sun lamps will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However for modelling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Scattering Bounces

Real world effects such as scattering in clouds or subsurface scattering require many scattering bounces. However unbiased rendering of such effects is slow and noisy. In typical movie production scenes only 0 or 1 bounces might be used to keep render times under control. The effect you get when rendering with 0 volume bounces is what is known as "single scattering", the effect from more bounces is "multiple scattering".

For rendering materials like skin or milk, the subsurface scattering shader is an approximation of such multiple scattering effects that is significantly more efficient but not as accurate.

For materials such as clouds or smoke that do not have a well defined surface, volume rendering is required. These look best with many scattering bounces, but in practice one might have to limit the number of bounces to keep render times acceptable.

Limitations

Currently we do not support:

General:

- Camera inside a volume mesh (Support added in Blender newer than 2.72).
- Correct ray visibility for volume meshes

On GPU:

- Smoke/Fire rendering
- Equi Angular / MIS Volume Sampling
- Volume Multi Light sampling

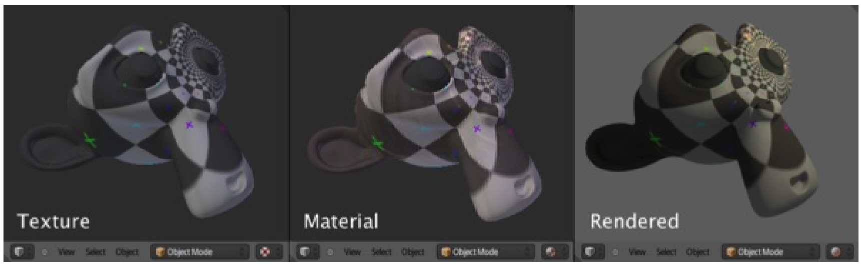
Texture Editing

3D viewport draw types, UV mapping, and texture painting work somewhat differently when Cycles is enabled. UV Maps no longer get image textures assigned themselves; rather they must always be assigned by adding an image texture node to a material.

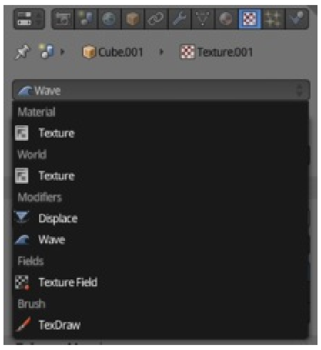
3D Viewport Draw Types

The Texture draw types used for Blender Internal have been replaced by three others in Cycles:

- *Texture*: this draw mode is used for editing, painting and mapping individual textures. Lighting is the same as in solid mode, so this is similar to the existing textured solid for Blender Internal. The texture drawn is the active image texture node for the material.
- *Material*: a simplified version of the entire material is drawn using GLSL shaders. This uses solid lighting, and also is mostly useful for editing, painting and mapping textures, but while seeing how they integrate with the material.
- *Rendered*: in this draw mode the render engine does the drawing, interactively refining the full rendered image by taking more samples. Unlike offline rendering, objects still use the viewport rather than render resolution and visibility.



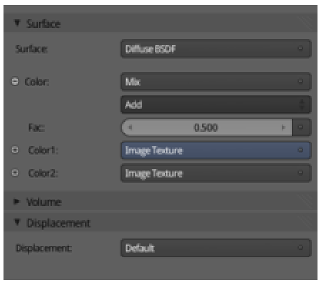
Texture Properties



In the texture properties, the texture can now be selected from a list that contains all texture nodes from the world, lamps and materials, but also from e.g. modifiers, brushes and physics fields.

For shading nodes, the available textures are Cycles textures. For others, Blender textures are still used, but this will change in the future.

Painting & UV Editing



For texture paint mode, the image that is painted on is taken from the slot panel in the toolbar. Selecting a slot also sets the active image texture.

For UV mapping, the active UV map as specified in the mesh properties is used. Assigning images in the image editor also affects the active image texture node.

World

Lighting with an HDR image

The world environment can emit light, ranging from a single solid color, physical sky model, to arbitrary textures.

Surface Shader

The surface shader defines the light emission from the environment into the scene. The world surface is rendered as if it is very distant from the scene, and as such there is no two-way interacting between objects in the scene and the environment, only light coming in. The only shader accepted is the Background node with a color input and strength factor for the intensity of the light.

Image Based Lighting

For image based lighting, use the Environment Texture node rather than the Image Texture node for correct mapping. This supports Equirectangular (also known as Lat/Long) for environment maps, and Mirror Ball mapping for converting photos of mirror balls to environment maps.

Volume Shader

A volume shader can be applied to the entirely world, filling the entire space.

Currently this is most useful for night time or other dark scenes, as the world surface shader or sun lamps will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However for modelling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Ambient Occlusion

Ambient occlusion is a lighting method based on how much a point on a surface is occluded by nearby surfaces. This is a trick that is not physically accurate, but it is useful to emphasize shapes of surfaces, or as a cheap way to get an effect that looks a bit like indirect lighting.

- Factor: The strength of the ambient occlusion; value 1.0 is like a white world shader.
- Distance: Distance from shading point to trace rays. A shorter distance emphasizes nearby features, while longer distances make it also take objects further away into account.

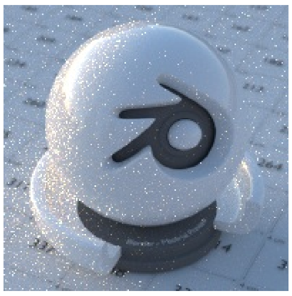
Lighting from ambient occlusion is only applied to diffuse reflection BSDFs; glossy or transmission BSDFs are not affected. Transparency of surfaces will be taken into account, i.e. a half-transparent surface will only half occlude.

An alternative method of using Ambient Occlusion on a per-shader basis is to use the Ambient Occlusion shader (*non-shader AO node still to be implemented*).

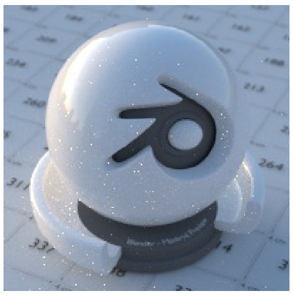
Settings

- Multiple Importance Sample: Enabling this will sample the background texture such that lighter parts are favored, producing less noise in the render. It is almost always a good idea to enable this when using an image texture to light the scene, otherwise noise can take a very long time to converge.
- Map Resolution: Sets the resolution of the 'Multiple Importance Sample' map. Higher values may produce less noise when using high-res images, but will take up more memory and render slightly slower.

Below is a comparison between Multiple Importance Sample Off and On - both images rendered for 25 seconds (Off: 1500 samples, On: 1000 samples)



Multiple Importance Sample Off



Multiple Importance Sample On

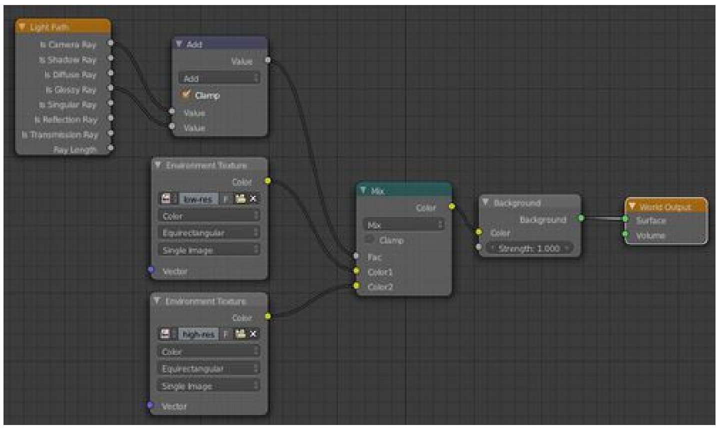
Ray Visibility

As with other objects, *Ray Visibility* allows you to control which other shaders can "see" the environment.

Tricks

Sometimes it may be useful to have a different background that is directly visible versus one that is indirectly lighting the objects. A simple solution to this is to add a Mix node, with the Blend Factor set to Is Camera Ray. The first input color is then the indirect color, and the second the directly visible color. This is useful when using a high-res image for the background and a low-res image for the actual lighting.

Similarly, adding the *Is Camera* and *Is Glossy* rays will mean that the high-res image will also be visible in reflections.



Nodes for the trick above

Lamps

Next to lighting from the background and any object with an emission shader, lamps are another way to add light into the scene. The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

Type
: Currently **Point**, **Spot**, **Area** and **Sun** lamps are supported. Hemi lamps are not supported, and will be rendered as point and sun lamps respectively, but they may start working in the future, so it's best not to enable them to preserve compatibility.

Size
: Size of the lamp in Blender Units; increasing this will result in softer shadows and shading.

Cast Shadow
: By disabling this option, light from lamps will not be blocked by objects in-between. This can speed up rendering by not having to trace rays to the light source.

Point Lamp

Point lamps emit light equally in all directions. By setting the Size larger than zero, they become spherical lamps, which give softer shadows and shading. The strength of point lamps is specified in Watts.

Spot Lamp

Spot lamps emit light in a particular direction, inside a cone. By setting the Size larger than zero, they can cast softer shadows and shading. The size parameter defines the size of the cone, while the blend parameter can soften the edges of the cone.

Area Lamp

Area lamps emit light from a square or rectangular area with a Lambertian distribution.

Sun Lamp

Sun lamps emit light in a given direction. Their position is not taken into account; they are always located outside of the scene, infinitely far away, and will not result in any distance falloff.

Because they are not located inside the scene, their strength uses different units, and should typically be set to lower values than other lights.

Nodes

Materials, lights and backgrounds are all defined using a network of shading nodes. These nodes output values, vectors, colors and shaders.

Shaders

An important concept to understand when building node setups is that of the **shader socket**. The output of all surface and volume shaders is a shader, describing lighting interaction at the surface or of the volume, rather than the color of the surface.

There are a few types of shaders available as nodes:

- **BSDF** shader describing light reflection, refraction and absorption at an object surface.
- **Emission** shader describing light emission at an object surface or in a volume.
- **Volume** shader describing light scattering inside a volume.
- **Background** shader describing light emission from the environment.

Each shader node has a color input, and outputs a shader. These can then be mixed and added together using Mix and Add Shader nodes. No other operations are permitted. The resulting output can then be used by the render engine to compute all light interactions, for direct lighting or global illumination.

Textures

Each texture type in Cycles corresponds to a node, with a texture coordinate and various parameters as input, and a color or value as output. No texture datablocks are needed; instead node groups can be used for reusing texture setups.

For UV mapping and texture painting in the viewport, the Image texture node must be used. When setting such a node as active, it will be drawn in Textured draw mode, and can be painted on in texture paint mode.

The default texture coordinates for all nodes are Generated coordinates, with the exception of Image textures that use UV coordinates by default. Each node includes some options to modify the texture mapping and resulting color, and these can be edited in the texture properties.

More

Nodes for geometric data, texture coordinates, layering shaders and non-physically based tricks.

Open Shading Language

Custom nodes can be written using the Open Shading Language.

Input Nodes

Camera Data

View Vector
: A Camera space vector from the camera to the shading point.

View Z Depth

View Distance
: Distance from the camera to the shading point.

Value

Input a scalar value.

Value
: Value output.

RGB

Input an RGB color.

Color
: RGB color output.

Attribute

Retrieve attribute attached to the object or mesh. Currently UV maps and vertex color layers can be retrieved this way by their names, with layers and attributes planned to be added. Also internal attributes like *P* (position), *N* (normal), *Ng* (geometric normal) may be accessed this way, although there are more convenient nodes for this.

Name
: Name of the attribute.

Color output
: RGB color interpolated from the attribute.

Vector output
: XYZ vector interpolated from the attribute.

Fac output
: Scalar value interpolated from the attribute.

Geometry

Geometric information about the current shading point. All vector coordinates are in *World Space*. For volume shaders, only the position and incoming vector are available.

Position
: Position of the shading point.

Normal
: Shading normal at the surface (includes smooth normals and bump mapping).

Tangent
: Tangent at the surface.

True Normal
: Geometry or flat normal of the surface.

Incoming
: Vector pointing towards the point the shading point is being viewed from.

Parametric
: Parametric coordinates of the shading point on the surface.

Backfacing
: 1.0 if the face is being viewed from the backside, 0.0 for the frontside.

Light Path

Node to find out for which kind of incoming ray the shader is being executed; particularly useful for non-physically based tricks. More information about the meaning of each type is in the Light Paths documentation.

Is Camera Ray output
: 1.0 if shading is executed for a camera ray, 0.0 otherwise.

Is Shadow Ray output
: 1.0 if shading is executed for a shadow ray, 0.0 otherwise.

Is Diffuse Ray output
: 1.0 if shading is executed for a diffuse ray, 0.0 otherwise.

Is Glossy Ray output
: 1.0 if shading is executed for a glossy ray, 0.0 otherwise.

Is Singular Ray output
: 1.0 if shading is executed for a singular ray, 0.0 otherwise.

Is Reflection Ray output
: 1.0 if shading is executed for a reflection ray, 0.0 otherwise.

Is Transmission Ray output

1.0 if shading is executed for a transmission ray, 0.0 otherwise.

Ray Length output
: Distance traveled by the light ray from the last bounce or camera.

Object Info

Information about the object instance. This can be useful to give some variation to a single material assigned to multiple instances, either manually controlled through the object index, based on the object location, or randomized for each instance. For example a Noise texture can give random colors or a Color ramp can give a range of colors to be randomly picked from.

Location
: Location of the object in world space.

Object Index
: Object pass index, same as in the Object Index pass.transformed.

Material Index
: Material pass index, same as in the Material Index pass.

Random
: Random number unique to a single object instance.

Fresnel

Dielectric fresnel, computing how much light is refracted through and how much is reflected off a layer. The resulting weight can be used for layering shaders with the Mix Shader node. It is dependent on the angle between the surface normal and the viewing direction.

IOR input
: Index of refraction of the material being entered.

Fresnel output
: Fresnel weight, indicating the probability with which light will reflect off the layer rather than passing through.

Layer Weight

Output weights typically used for layering shaders with the Mix Shader node.

Blend input
: Blend between the first and second shader.

Fresnel output
: Dielectric fresnel weight, useful for example for layering diffuse and glossy shaders to create a plastic material. This is like the Fresnel node, except that the input of this node is in the often more-convenient 0.0 to 1.0 range.

Facing output
: Weight that blends from the first to the second shader as the surface goes from facing the viewer to viewing it at a grazing angle.

Texture Coordinates

Commonly used texture coordinates, typically used as inputs for the Vector input for texture nodes.

Generated
: Automatically generated texture coordinates from the vertex positions of the mesh without deformation, keeping them sticking to the surface under animation. Range from 0.0 to 1.0 over the bounding box of the undeformed mesh.

Normal
: Object space normal, for texturing objects with the texture staying fixed on the object as it transforms.

UV
: UV texture coordinates from the active render UV layer.

Object
: Position coordinate in object space.

Camera
: Position coordinate in camera space.

Window
: Location of shading point on the screen, ranging from 0.0 to 1.0 from the left to right side and bottom to top of the render.

Reflection
: Vector in the direction of a sharp reflection; typically used for environment maps.

Particle Info

For objects instanced from a particle system, this node give access to the data of the particle that spawned the instance.

Index
: Index number of the particle (from 0 to number of particles).

Age
: Age of the particle in frames.

Lifetime
: Total lifespan of the particle in frames.

Location
: Location of the particle.

Size
: Size of the particle.

Velocity

Velocity of the particle.

Angular Velocity
: Angular velocity of the particle.

Hair Info

This node gives access to strand information.

Is strand
: Returns 1 when the shader is acting on a strand, otherwise 0.

Intersect
: The point along the strand where the ray hits the strand (1 at the tip and 0 at the root).

Thickness
: The thickness of the strand at the point where the ray hits the strand.

Tangent Normal
: Tangent normal of the strand.

Tangent

Generates a tangent direction for the Anisotropic BSDF.

Direction Type
: The tangent direction can be derived from a cylindrical projection around the X, Y or Z axis (Radial), or from a manually created UV Map for full control.

Tangent Output
: The tangent direction vector.

Output Nodes

Output nodes are the final node in every node tree. Although you can add more than one, only one will be used (indicated by a colored or darkened header). Output nodes are always preceded by Shaders except in the case of the Displacement of a Material Output.

Material Output

- Surface: The surface output of the material
- Volume:*Currently under independent development, does nothing*
- Displacement: Used to create bump mapping or actual subdivided Displacement

Lamp Output

- Surface: Not an actual surface, but the final output of a Lamp Object

World Output

- Surface: The appearance of the environment, usually preceded by a Background Shader
- Volume:*Currently under independent development, does nothing*

Shader Nodes

BSDF

Diffuse

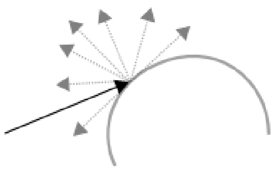
Lambertian and Oren-Nayar diffuse reflection.

Color input
: Color of the surface, or physically speaking, the probability that light is reflected or transmitted for each wavelength.

Roughness input
: Surface roughness; 0.0 gives standard Lambertian reflection, higher values activate the Oren-Nayar BSDF.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Diffuse BSDF shader.


Diffuse behavior

Lambertian Diffuse (Roughness=0)

Oren Nayar Diffuse (Roughness=1)

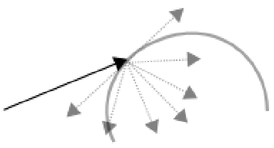
Translucent

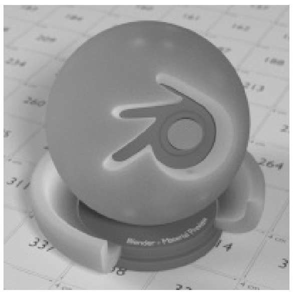
Lambertian diffuse transmission.

Color input
: Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Translucent BSDF shader.




Translucent Shader

Glossy

Glossy reflection with microfacet distribution, used for materials such as metal or mirrors.

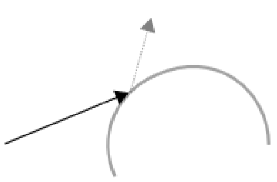
Distribution
: Microfacet distribution to use. Sharp results in perfectly sharp reflections like a mirror, while Beckmann, GGX and Ashikhmin-Shirley can use the Roughness input for blurry reflections.

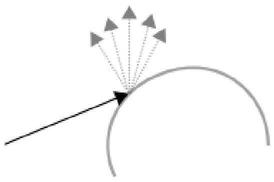
Color input
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness input
: Influences sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

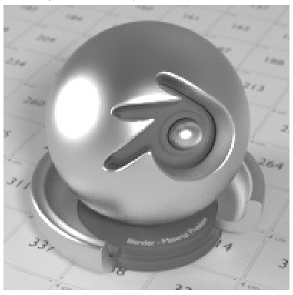
Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Glossy BSDF shader.


Sharp Glossy behavior


Rough Glossy behavior

A Sharp Glossy Material


A Rough Glossy Material

Anisotropic

Anisotropic glossy reflection, with separate control over U and V direction roughness. The tangents used for shading are derived from the active UV map. If no UV map is available, they are automatically generated using a sphere mapping based on the mesh bounding box.

Distribution
: Microfacet distribution to use. Sharp results in perfectly sharp reflections like a mirror, while Beckmann, GGX and Ashikhmin-Shirley can use the Roughness input for blurry reflections.

Color input
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness input
: Sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

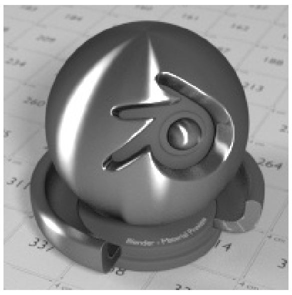
Anisotropy input
: Amount of anisotropy in the reflection; 0.0 gives a round highlight. Higher values give elongated highlights orthogonal to the tangent direction; negative values give highlights shaped along the tangent direction.

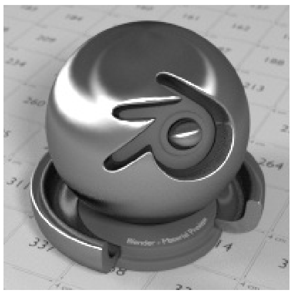
Rotation input
: Rotation of the anisotropic tangent direction. Value 0.0 equals 0° rotation, 0.25 equals 90° and 1.0 equals 360° = 0°. This can be used to texture the tangent direction.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

Tangent input
: Tangent used for shading; if nothing is connected the default shading tangent is used.

BSDF output
: Anisotropic glossy BSDF shader.


Anisotropic rotation on 0


Anisotropic rotation on 0.25 (90°)

Toon

Diffuse and Glossy Toon BSDF for creating cartoon light effects.

Color input
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Size input
: Parameter between 0.0 and 1.0 that gives a angle of reflection between 0° and 90°.

Smooth input

This value specifies an angle over which a smooth transition from full to no reflection happens.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Toon BSDF shader.

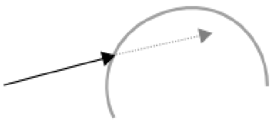
Toon Shader

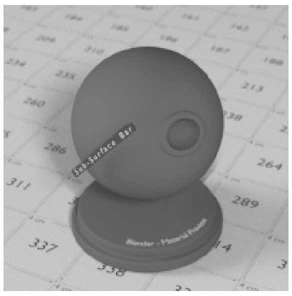
Transparent

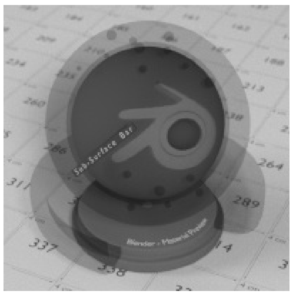
Transparent BSDF without refraction, passing straight through the surface, as if there were no geometry there. Useful with alpha maps, for example. This shader affects light paths somewhat differently than other BSDFs. Note that only pure white transparent shaders are completely transparent.

Color input
: Color of the surface, or physically speaking, the probability for each wavelength that light is blocked or passes straight through the surface.

BSDF output
: Transparent BSDF shader.


Transparent behaviour


Transparent Shader (pure white)


Transparent Shader (gray)

Glass

Glass-like shader mixing refraction and reflection at grazing angles. Like the transparent shader, only pure white will make it transparent. The glass shader tends to cause noise due to caustics. Since the Cycles path tracing integrator is not very good at rendering caustics, it helps to combine this with a transparent shader for shadows; for more details see here.

Distribution
: Microfacet distribution to use. Sharp results in perfectly sharp refractions like clear glass, while Beckmann and GGX can use the Roughness input for rough glass.

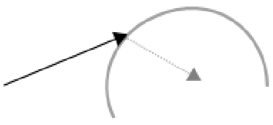
Color input
: Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

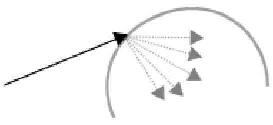
Roughness input
: Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

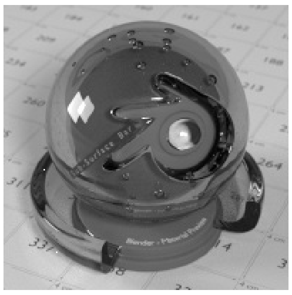
IOR input
: Index of refraction defining how much the ray changes direction. At 1.0 rays pass straight through like transparent; higher values give more refraction.

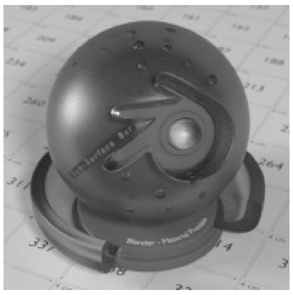
Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Glass BSDF shader.


Sharp Glass behaviour


Rough Glass behaviour


A Sharp Glass Material


A Rough Glass Material

Refraction

Glossy refraction with sharp or microfacet distribution, used for materials that transmit light. For best results this node should be considered as a building block and not be used on its own, but rather mixed with a glossy node using a fresnel factor. Otherwise it will give quite dark results at the edges for glossy refraction.

Distribution
: Microfacet distribution to use. Sharp results in perfectly sharp refractions, while Beckmann and GGX can use the Roughness input for blurry refractions.

Color input
: Color of the surface, or physically speaking, the probability that light is refracted for each wavelength.

Roughness input
: Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Glossy BSDF shader.

Refraction Shader.

Velvet

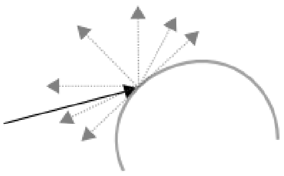
Velvet reflection shader for materials such as cloth. It is meant to be used together with other shaders (such as a *Diffuse Shader*) and isn't particularly useful on it's own.

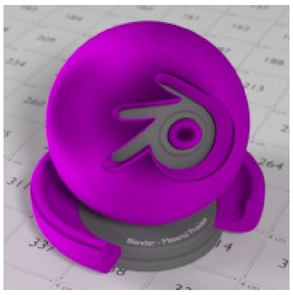
Color input
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Sigma input
: Variance of the normal distribution, controlling the sharpness of the peak - can be thought of as a kind of *roughness*.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

BSDF output
: Velvet BSDF shader.




The Velvet Shader

BSSRDF

Subsurface Scattering

Simple subsurface multiple scattering, for materials such as skin, wax, marble, milk and others. For these materials, rather than light being reflect directly off the surface, it will penetrate the surface and bounce around internally before getting absorbed or leaving the surface at a nearby point.

How far the color scatters on average can be configured per RGB color channel. For example, for skin, red colors scatter further, which gives distinctive red-colored shadows, and a soft appearance.

Falloff
: Lighting distance falloff function.
: **Cubic** is a sharp falloff useful for many simple materials. The function is (radius - x)^3
: **Gaussian** gives a smoother falloff following a normal distribution, which is particularly useful for more advanced materials that use measured data that was fitted to one or more such Gaussian functions. The function is $e^{-8x^2/radius^2}$, such that the radius roughly matches the maximum falloff distance. To match a given measured variance v, set radius = sqrt(16*v).

Color input
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Scale input
: Global scale factor for the scattering radius.

Radius input
: Scattering radius for each RGB color channel, the maximum distance that light can scatter.

Normal input
: Normal used for shading; if nothing is connected the default shading normal is used.

Texture Blur input
: How much of the texture will be blurred along with the lighting, mixing the texture at the incoming and outgoing points on the surface. Note that the right choice depends on the texture. Consider for example a texture created from a photograph of skin, in this cases the colors will already be pre-blurred and texture blur could be set to 0. Even for hand painted textures no or minimal blurring might be appropriate, as a texture artist would likely paint in softening already, one would usually not even know what an unblurred skin texture looks like, we always see it blurred. For a procedural texture on the other hand this option would likely have a higher value.

BSSRDF output
: BSSRDF shader.

A skin-toned SSS shader with color radius: 1.0, 0.8, 0.5.

Emission

Lambertian emission, to be used for material and lamp surface outputs.

Color input
: Color of the emitted light.

Strength input
: Strength of the emitted light. For point and area lamps, the unit is Watts. For materials, a value of 1.0 will ensure that the object in the image has the exact same color as the Color input, i.e. make it 'shadeless'.

Emission output
: Emission shader.

A white Emissive material, with strength at 1.0.

A white Emissive material, with strength at 3.0.

Cycles uses a physically correct light falloff by default, whereas Blender Internal uses a smoothed falloff with a Distance parameter. A similar effect can be found by using the Light Falloff node with the Smooth parameter.

Lamp strength for point, spot and area lamps is specified in Watts. This means you typically need higher values than Blender Internal, as you couldn't use a 1W lamp to light a room; you need something stronger like a 100W lamp.

Sun lamps are specified in Watts/m^2, which require much smaller values like 1 W/m^2. This can be confusing, but specifying strength in Watts wouldn't have been convenient; the real sun for example has strength 384600000000000000000000000W. Emission shaders on meshes are also in Watts/m^2.

Background

Background light emission. This node should only be used for the world surface output; it is ignored in other cases.

Color input
: Color of the emitted light.

Strength input
: Strength of the emitted light.

Background output
: Background shader.

Holdout

A holdout shader is useful for compositing, to create a "hole" in the image with zero alpha transparency where the object with this shader is located.

Holdout output
: Holdout shader.

The white area is a region with zero Alpha.

Ambient Occlusion

The ambient occlusion node gives per-material control for the amount of AO. When AO is enabled in the world, it affects all diffuse BSDFs in the scene. With this option it's possible to let only some materials be affected by AO, or to let it influence some materials more or less than others.

Color input
: surface reflection color.

AO output
: Ambient Occlusion shader.

White AO shader.

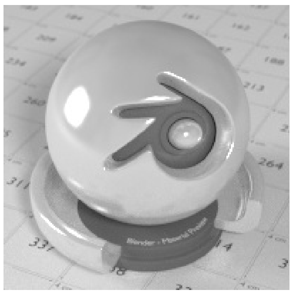
Mix and Add

Mix or add shaders together. Mixing can be used for material layering, where the Fac input may, for example, be connected to a Blend Weight node.

Shader inputs
: Shaders to mix, such that incoming rays hit either with the specified probability in the Fac socket.

Fac input
: Blend weight to use for mixing two shaders; at zero it uses the first shader entirely and at one the second shader.

Shader output
: Mixed shader.


A mix of a glossy and a diffuse shader makes a nice ceramic material.

Texture Nodes

Image Texture

Image texture from
GoodTextures.com

Use an image file as a texture.

Image Datablock
: Image datablock used as the image source. Currently not all images supported by Blender can be used by Cycles. In particular, generated, packed images or animations are not supported currently.

Projection
: Projection to use for mapping the textures. Flat will use the XY coordinates for mapping. Box will map the image to the 6 sides of a virtual box, based on the normal, using XY, YZ and XYZ coordinates depending on the side.

Projection Blend
: For Box mapping, the amount to blend between sides of the box, to get rid of sharp transitions between the different sides. Blending is useful to map a procedural-like image texture pattern seamlessly on a model. 0.0 gives no blending; higher values give a smoother transition.

Color Space
: Type of data that the image contains, either Color or Non-Color Data. For most color textures the default of Color should be used, but in case of e.g. a bump or alpha map, the pixel values should be interpreted as Non-Color Data, to avoid doing any unwanted color space conversions.

Vector input
: Texture coordinate for texture lookup. If this socket is left unconnected, UV coordinates from the active UV render layer are used.

Color output
: RGB color from image. If the image has alpha, the color is premultiplied with alpha if the Alpha output is used, and unpremultiplied or straight if the Alpha output is not used.

Alpha output
: Alpha channel from image.

Environment Texture

HDR image from
OpenFootage.net

Use an environment map image file as a texture. The environment map is expected to be in Latitude/Longitude or 'latlong' format.

Image Datablock
: Image datablock used as the image source. Currently not all images supported by Blender can be used by Cycles. In particular, generated, packed images or animations are not supported currently.

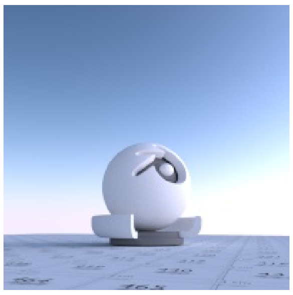
Color Space
: Type of data that the image contains, either Color or Non-Color Data. For most color textures the default of Color should be used, but in case of e.g. a bump or alpha map, the pixel values should be interpreted as Non-Color Data, to avoid doing any unwanted color space conversions.

Vector input
: Texture coordinate for texture lookup. If this socket is left unconnected, the image is mapped as environment with the Z axis as up.

Color output
: RGB color from the image. If the image has alpha, the color is premultiplied with alpha if the Alpha output is used, and unpremultiplied if the Alpha output is not used.

Alpha output
: Alpha channel from image.

Sky Texture


Sky Texture

Procedural Sky texture.

Sky Type
: Sky model to use (Preetham or Hosek / Wilkie).

Sun Direction
: Sun direction vector.

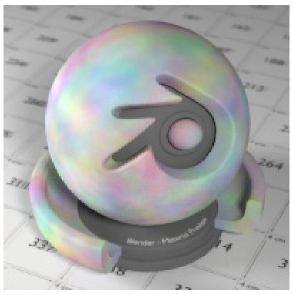
Turbidity
: Atmospheric turbidity. (2: Arctic like, 3: clear sky, 6: warm/moist day, 10: hazy day)

Ground Albedo
: Amount of light reflected from the planet surface back into the atmosphere. (RGB 0,0,0 is black, 1,1,1 is white).

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Color output
: Texture color output.

Noise Texture


Noise Texture with high detail

Procedural Perlin noise texture, similar to the Clouds texture in Blender Internal.

Vector input
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale input
: Overall texture scale.

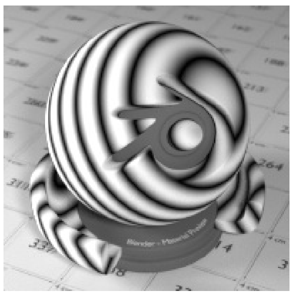
Detail input
: Amount of noise detail.

Distortion input
: Amount of distortion.

Color output
: Texture color output.

Fac output
: Texture intensity output.

Wave Texture


Default wave texture

Procedural bands or rings texture with noise distortion.

Type
: Bands or Rings shaped waves.

Vector input
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale input
: Overall texture scale.

Distortion input
: Amount of distortion of the wave (similar to the Marble texture in Blender Internal).

Detail input
: Amount of distortion noise detail.

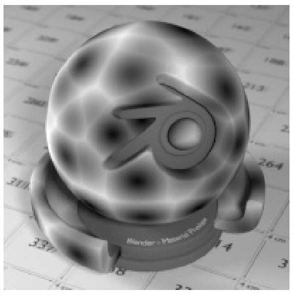
Detail Scale input
: Scale of distortion noise.

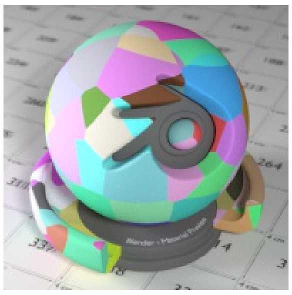
Color output
: Texture color output.

Fac output
: Texture intensity output.

Voronoi Texture

Procedural texture producing Voronoi cells.


Voronoi texture, type: Intensity


Voronoi texture, type: Cells

Type
: Intensity or Cells output.

Vector input
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

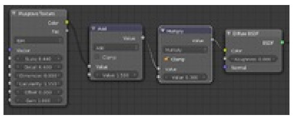
Scale input
: Overall texture scale.

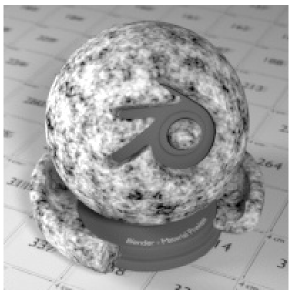
Color output
: Texture color output.

Fac output
: Texture intensity output.

Musgrave Texture

Advanced procedural noise texture. Note that it often needs some adjustments (multiplication and addition) in order to see more detail.


Nodes for the image to the right


Remapped Musgrave texture such that most values are visible

Type
: Multifractal, Ridged Multifractal, Hybrid Multifractal, fBM, Hetero Terrain.

Vector input
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale input
: Overall texture scale.

Detail input
: Amount of noise detail.

Dimension input
: *TBD*

Lacunarity input
: *TBD*

Offset input
: *TBD (Does not affect Multifractal and fBM)*

Gain input
: *TBD (Does not affect Multifractal and fBM)*

Color output
: Texture color output.

Fac output
: Texture intensity output.

Gradient Texture

Gradient texture using object coordinates

A gradient texture.

Type
: The gradient can be Linear, Quadratic, Easing, Diagonal, Spherical, Quadratic Sphere or Radial.

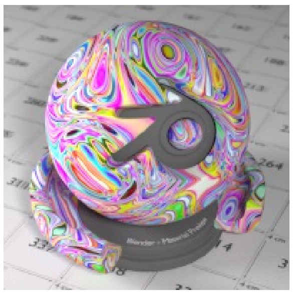
Vector input

Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Color output
: Texture color output.

Fac output
: Texture intensity output.

Magic Texture


Magic texture: Depth 10, Distortion 2.0

Psychedelic color texture.

Depth
: Number of iterations.

Vector input
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Distortion input
: Amount of distortion.

Color output
: Texture color output.

Fac output
: Texture intensity output.

Checker Texture

Default Checker texture

Checkerboard texture.

Vector input
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Color1/2 input
: Color of the checkers.

Scale input
: Overall texture scale.

Color output
: Texture color output.

Fac output
: Checker 1 mask (1 = Checker 1).

Brick Texture

Brick texture: Colors changed, Squash 0.62, Squash Frequency 3.

Procedural texture producing Bricks.

Options

Offset
: Determines the brick offset of the various rows.

Frequency
: Determines the offset frequency. A value of 2 gives a even/uneven pattern of rows.

Squash
: Amount of brick squashing.

Frequency
: Brick squashing frequency.

Sockets

Color 1/2 and Mortar
: Color of the bricks and mortar.

Scale
: Overall texture scale.

Mortar Size
: The Mortar size; 0 means no Mortar.

Bias
: The color variation between Brick color 1 / 2. Values of -1 and 1 only use one of the two colors; values in between mix the colors.

Brick Width
: The width of the bricks.

Row Height
: The height of the brick rows.

Color output
: Texture color output.

Fac output
: Mortar mask (1 = mortar).

Vector Nodes

Vector Transform

Vector Curves

Normal

Normal Map

Bump

Mapping

Convertor Nodes

Blackbody

Wavelength

Separate HSV

Combine HSV

Separate RGB

Vector Math

RGB to BW

ColorRamp

Math

Open Shading Language

Users can now create their own nodes using the Open Shading Language (OSL). Note that these nodes will only work for CPU rendering; there is no support for running OSL code on the GPU. **To enable it, select Open Shading Language as the shading system** in the render settings.

Note: on Linux, C/C++ compiler tools (in particular /usr/bin/cpp) must be installed to compile OSL scripts.

Script Node

OSL was designed for node-based shading, and **each OSL shader corresponds to a node** in a node setup. To add an OSL shader, add a script node and link it to a text datablock or an external file. Input and output sockets will be created from the shader parameters on clicking the update button in the node or the text editor.

OSL shaders can be linked to the node in a few different ways. With the **Internal** mode, a text datablock is used to store the OSL shader, and the OSO bytecode is stored in the node itself. This is useful for distributing a .blend file with everything packed into it.

The **External** mode can be used to specify a .osl file on disk, and this will then be automatically compiled into a .oso file in the same directory. It is also possible to specify a path to a .oso file, which will then be used directly, with compilation done manually by the user. The third option is to specify just the module name, which will be looked up in the shader search path.

The shader search path is located in the same place as the scripts or configuration path, under:

Linux	`/home/$user/.config/blender/`*`Version Number`*`/shaders/`
Windows	`C:\Users\$user\AppData\Roaming\Blender Foundation\Blender\`*`Version Number`*`\shaders\`
Mac OS X	`/Users/$user/Library/Application Support/Blender/`*`Version Number`*`/shaders/`

(Replace *Version Number* with the release number of your current Blender installation, e.g. *2.65* or *2.66*.)

For use in production, we suggest to **use a node group to wrap shader script nodes**, and link that into other .blend files. This makes it easier to make changes to the node afterwards as sockets are added or removed, without having to update the script nodes in all files.

Writing Shaders

For more details on how to write shaders, see the OSL specification. Here is a simple example:

```
shader simple_material(
    color Diffuse_Color = color(0.6, 0.8, 0.6),
    float Noise_Factor = 0.5,
    output closure color BSDF = diffuse(N))
{
   color material_color = Diffuse_Color * mix(1.0, noise(P * 10.0), Noise_Factor);
   BSDF = material_color * diffuse(N);
}
```

Closures

OSL is different from, for example, RSL or GLSL, in that it does not have a light loop. There is no access to lights in the scene, and the material must be built from closures that are implemented in the render engine itself. This is more limited, but also makes it possible for the render engine to do optimizations and ensure all shaders can be importance sampled.

The available closures in Cycles correspond to the shader nodes and their sockets; for more details on what they do and the meaning of the parameters, see the shader nodes manual.

BSDF

- `diffuse(N)`
- `oren_nayar(N, roughness)`
- `reflection(N)`
- `refraction(N, ior)`
- `microfacet_beckmann(N, roughness)`
- `microfacet_beckmann_refraction(N, roughness, ior)`
- `microfacet_ggx(N, roughness)`
- `microfacet_ggx_refraction(N, roughness, ior)`
- `phong_ramp(N, exponent, colors[8])`
- `diffuse_ramp(N, colors[8])`
- `translucent(N)`
- `transparent()`
- `ashikhmin_velvet(N, roughness)`
- `ward(N, T, roughness_u, roughness_v)`
- `diffuse_toon(N, size, smooth)`
- `glossy_toon(N, size, smooth)`
- `westin_sheen(N, roughness)`
- `westin_backscatter(N, edginess)`
- `hair_reflection(N, roughnessu, roughnessv, T, offset)`
- `hair_transmission(N, roughnessu, roughnessv, T, offset)`

BSSRDF

- `bssrdf_cubic(N, radius, texture_blur, sharpness)`
- `bssrdf_gaussian(N, radius, texture_blur)`

Volume

- `henyey_greenstein(g)` New in version 2.70
- `absorption()` New in version 2.70

Other

- `emission()`
- `ambient_occlusion()`
- `holdout()`
- `background()`

Attributes

Some object, particle and mesh attributes are available to the built-in getattribute() function. UV maps and vertex colors can be retrieved using their name. Other attributes are listed below:

`geom:generated`	Generated texture coordinates
`geom:uv`	Default render UV map
`geom:dupli_generated`	For instances, generated coordinate from duplicator object
`geom:dupli_uv`	For instances, UV coordinate from duplicator object
`geom:trianglevertices`	3 vertex coordinates of the triangle
`geom:numpolyvertices`	Number of vertices in the polygon (always returns 3 currently)
`geom:polyvertices`	Vertex coordinates array of the polygon (always 3 vertices currently)
`geom:name`	Name of the object
`geom:is_curve`	Is object a strand or not
`geom:curve_intercept`	Point along the strand, from root to tip
`geom:curve_thickness`	Thickness of the strand
`geom:curve_tangent_normal`	Tangent Normal of the strand
`path:ray_length`	Ray distance since last hit
`object:location`	Object location
`object:index`	Object index number
`object:random`	Per object random number generated from object index and name
`material:index`	Material index number
`particle:index`	Particle instance number
`particle:age`	Particle age in frames
`particle:lifetime`	Total lifespan of particle in frames
`particle:location`	Location of the particle
`particle:size`	Size of the particle
`particle:velocity`	Velocity of the particle
`particle:angular_velocity`	Angular velocity of the particle

Trace

We support the trace(point pos, vector dir, ...) function, to trace rays from the OSL shader. The "shade" parameter is not supported currently, but attributes can be retrieved from the object that was hit using the getmessage("trace", ..) function. See the OSL specification for details on how to use this.

This function can't be used instead of lighting; the main purpose is to allow shaders to "probe" nearby geometry, for example to apply a projected texture that can be blocked by geometry, apply more “wear” to exposed geometry, or make other ambient occlusion-like effects.

More Nodes

Value

Input a scalar value.

Value
: Value output.

RGB

Input an RGB color.

Color
: RGB color output.

Geometry

Geometric information about the current shading point. All vector coordinates are in *World Space*. For volume shaders, only the position and incoming vector are available.

Position
: Position of the shading point.

Normal
: Shading normal at the surface (includes smooth normals and bump mapping).

Tangent
: Tangent at the surface.

True Normal
: Geometry or flat normal of the surface.

Incoming
: Vector pointing towards the point the shading point is being viewed from.

Parametric
: Parametric coordinates of the shading point on the surface.

Backfacing
: 1.0 if the face is being viewed from the backside, 0.0 for the frontside.

Wireframe

Node for a wireframe shader (Triangles only for now).

Pixel Size
: Use screen pixel size instead of world units.

Size
: Controls the thickness of the wireframe.

Fac output
: 1.0 if shading is executed on an edge, 0.0 otherwise.

Wavelength

A wavelength to rgb converter.

Wavelength
: The color wavelength from 380 to 780 nanometers.

Color
: RGB color output.

Blackbody

A blackbody temperature to RGB converter.

Temperature
: The temperature in Kelvin.

Color
: RGB color output.

Texture Coordinates

Commonly used texture coordinates, typically used as inputs for the Vector input for texture nodes.

Generated
: Automatically-generated texture coordinates from the vertex positions of the mesh without deformation, keeping them sticking to the surface under animation. Range from 0.0 to 1.0 over the bounding box of the undeformed mesh.

Normal
: Object space normal, for texturing objects with the texture staying fixed on the object as it transformed.

UV
: UV texture coordinates from the active render UV layer.

Object

Position coordinate in object space.

Camera
: Position coordinate in camera space.

Window
: Location of shading point on the screen, ranging from 0.0 to 1.0 from the left to right side and bottom to top of the render.

Reflection
: Vector in the direction of a sharp reflection, typically used for environment maps.

Bump

Generate a perturbed normal from a height texture, for bump mapping. The height value will be sampled at the shading point and two nearby points on the surface to determine the local direction of the normal.

Invert
: Invert the bump mapping, to displace into the surface instead of out.

Strength Input
: Strength of the bump mapping effect, interpolating between no bump mapping and full bump mapping.

Distance Input
: Multiplier for the height value to control the overall distance for bump mapping.

Height Input
: Scalar value giving the height offset from the surface at the shading point; this is where you plug in textures.

Vector Transform

Allows converting a Vector, Point or Normal between World <=> Camera <=> Object coordinate space.

Type
: Specifies the input/output type: Vector, Point or Normal.

Convert From
: Coordinate Space to convert from: World, Object or Camera.

Convert To
: Coordinate Space to convert to: World, Object or Camera.

Vector Input
: The input vector.

Vector Output
: The transformed output vector.

Tangent

Generate a tangent direction for the Anisotropic BSDF.

Direction Type
: The tangent direction can be derived from a cylindrical projection around the X, Y or Z axis (Radial), or from a manually created UV Map for full control.

Tangent Output
: The tangent direction vector.

Normal Map

Generate a perturbed normal from an RGB normal map image. This is usually chained with an Image Texture node in the color input, to specify the normal map image. For tangent space normal maps, the UV coordinates for the image must match, and the image texture should be set to Non-Color mode to give correct results.

Space
: The input RGB color can be in one of 3 spaces: Tangent, Object and World space. Tangent space normal maps are the most common, as they support object transformation and mesh deformations. Object space normal maps keep sticking to the surface under object transformations, while World normal maps do not.

UV Map
: Name of the UV map to derive normal mapping tangents from. When chained with an Image Texture node, this UV map should be the same as the UV map used to map the texture.

Strength
: Strength of the normal mapping effect.

Color Input
: RGB color that encodes the normal in the specified space.

Normal Output
: Normal that can be used as an input to BSDF nodes.

Object Info

Information about the object instance. This can be useful to give some variation to a single material assigned to multiple instances, either manually controlled through the object index, based on the object location, or randomized for each instance. For example a Noise texture can give random colors or a Color ramp can give a range of colors to be randomly picked from.

Note that this node only works for material shading nodes; it does nothing for lamp and world shading nodes.

Location
: Location of the object in world space.

Object Index
: Object pass index, same as in the Object Index pass.transformed.

Material Index
: Material pass index, same as in the Material Index pass.

Random
: Random number unique to a single object instance.

Particle Info

For objects instanced from a particle system, this node give access to the data of the particle that spawned the instance. This node currently only supports parent particles, info from child particles is not available.

Index
: Index number of the particle (from 0 to number of particles).

Age
: Age of the particle in frames.

Lifetime
: Total lifespan of the particle in frames.

Location
: Location of the particle.

Size
: Size of the particle.

Velocity
: Velocity of the particle.

Angular Velocity
: Angular velocity of the particle.

Hair Info

This node gives access to strand information.

Is strand
: Returns 1 when the shader is acting on a strand, otherwise 0.

Intersect
: The point along the strand where the ray hits the strand (1 at the tip and 0 at the root).

Thickness
: The thickness of the strand at the point where the ray hits the strand.

Tangent Normal
: Tangent normal of the strand.

Attribute

Retrieve attribute attached to the object or mesh. Currently UV maps and vertex color layers can be retrieved this way by their names, with layers and attributes planned to be added. Also internal attributes like *P* (position), *N* (normal), *Ng* (geometric normal) may be accessed this way, although there are more convenient nodes for this.

Name
: Name of the attribute.

Color output
: RGB color interpolated from the attribute.

Vector output
: XYZ vector interpolated from the attribute.

Fac output
: Scalar value interpolated from the attribute.

Mapping

Transform a coordinate; typically used for modifying texture coordinates.

Location
: Vector translation.

Rotation
: Rotation of the vector along XYZ axes.

Scale
: Scale of the vector.

Vector input
: Vector to be transformed.

Vector output
: Transformed vector.

Layer Weight

Output weights typically used for layering shaders with the Mix Shader node.

Blend input
: Blend between the first and second shader.

Fresnel output

Dielectric fresnel weight, useful for example to layer diffuse and glossy shaders to create a plastic material. This is like the Fresnel node, except that the input of this node is in the often more-convenient 0.0 to 1.0 range.

Facing output
: Weight that blends from the first to the second shader as the surface goes from facing the viewer to viewing it at a grazing angle.

Fresnel

Dielectric fresnel, computing how much light is reflected off a layer, where the rest will be refracted through the layer. The resulting weight can be used for layering shaders with the Mix Shader node. It is dependent on the angle between the surface normal and the viewing direction.

The most common use is to mix between two BSDFs using it as a blending factor in a mix shader node. For a simple glass material you would mix between a glossy refraction and glossy reflection. At grazing angles more light will be reflected than refracted as happens in reality.

For a two-layered material with a diffuse base and a glossy coating, you can use the same setup, mixing between a diffuse and glossy BSDF. By using the fresnel as the blending factor you're specifying that any light which is refracted through the glossy coating layer would hit the diffuse base and be reflected off that.

IOR input
: Index of refraction of the material being entered.

Fresnel output
: Fresnel weight, indicating the probability with which light will reflect off the layer rather than passing through.

Light Path

Node to find out for which kind of incoming ray the shader is being executed; particularly useful for non-physically based tricks. More information about the meaning of each type is in the Light Paths documentation.

Is Camera Ray output
: 1.0 if shading is executed for a camera ray, 0.0 otherwise.

Is Shadow Ray output
: 1.0 if shading is executed for a shadow ray, 0.0 otherwise.

Is Diffuse Ray output
: 1.0 if shading is executed for a diffuse ray, 0.0 otherwise.

Is Glossy Ray output
: 1.0 if shading is executed for a glossy ray, 0.0 otherwise.

Is Singular Ray output
: 1.0 if shading is executed for a singular ray, 0.0 otherwise.

Is Reflection Ray output
: 1.0 if shading is executed for a reflection ray, 0.0 otherwise.

Is Transmission Ray output
: 1.0 if shading is executed for a transmission ray, 0.0 otherwise.

Ray Length output
: Distance travelled by the light ray from the last bounce or camera.

Ray Depth output
: Returns the current light bounce.

Transparent Depth output
: Returns the number of transparent surfaces passed through.

Light Falloff

Manipulate how light intensity decreases over distance. In reality light will always fall off quadratically; however it can be useful to manipulate as a non-physically based lighting trick. Note that using Linear or Constant falloff may cause more light to be introduced with every global illumination bounce, making the resulting image extremely bright if many bounces are used.

Strength input
: Light strength before applying falloff modification.

Smooth input
: Smooth intensity of light near light sources. This can avoid harsh highlights, and reduce global illumination noise. 0.0 corresponds to no smoothing; higher values smooth more. The maximum light strength will be strength/smooth.

Quadratic output
: Quadratic light falloff; this will leave strength unmodified if smooth is 0.0 and corresponds to reality.

Linear output
: Linear light falloff, giving a slower decrease in intensity over distance.

Constant output
: Constant light falloff, where the distance to the light has no influence on its intensity.

Nodes shared with the Compositor

Some nodes are common with Composite nodes, their documentation can be found at their relevant pages rather than repeated here.

- Brightness Contrast
- Separate RGB
- Combine RGB
- Separate HSV
- Combine HSV
- Gamma

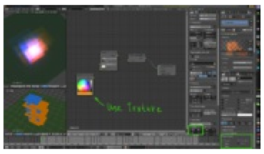
- Hue Saturation Value
- Invert
- Math
- Mix RGB
- RGB Curves
- RGB to BW
- Vector Curve

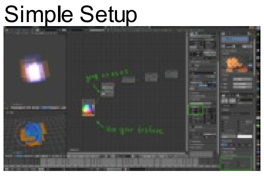
Cycles Additive Materials

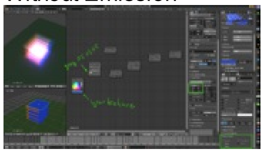
If you used game engines or other renderers. You know about Additive Blending for materials.

Also, the same material is in 3dMax. http://docs.garagegames.com/artist/official/3D%20Studio%20Max/AddiBlend.jpg

```
And here it is how it can be done in Cycles:
```


Simple Setup


Without Emission


With Emission

Another good additive
Material approach

Hair Rendering

Introduction to hair

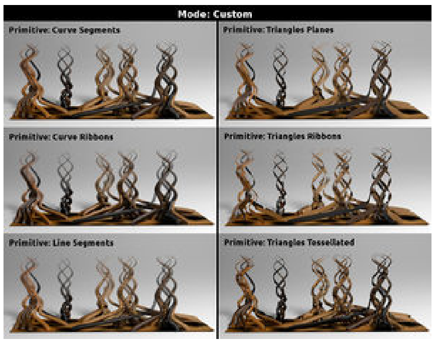
See the manual's hair documentation for details on controlling hair in Blender. The resolution of the strands is controlled by the step values in particle settings. Each hair system uses the material identified in the particle settings in the same way as Blender Internal.

Cycles Hair Rendering

The main settings for the strands can be found inside the *Cycles Hair Rendering* panel under the particle tab. There are a series of presets for strand rendering. The detailed options are available when *custom* is selected. This allows access to the following parameters:

Primitive

- Triangles uses a triangle mesh. There are three triangle options: Planes, Ribbons and Tessellated. Planes are triangles facing the camera. Ribbons are triangles following the strand direction. Tessellated generates a polygonal mesh, with the number of sides configurable.
- Line Segments uses a curve primitive. These are currently straight and create a perfect cylindrical segment between two points. A gap can appear between the cylinders. Tangent normals can be generated using the incoming ray and tangent vector. This tangent normal is the normal of a plane containing the tangent and inclined towards the incoming ray. Using tangent normals for shading can speed up convergence and reduce noise.
- Curve Segments uses a smooth Cardinal curve primitive. These interpolate a path through the curve keys. However, it renders slower than line segments. The interpolated path is subdivided to give points to connect. The parameter subdivisions sets the number of divisions used.
- Curve Ribbons uses a smooth Cardinal curve primitive. They are similar to curve segments but give a flat appearance.


Mode: Custom
Primitive: Curve Segments
Primitive: Triangles Planes
Primitive: Curve Ribbons
Primitive: Triangles Ribbons
Primitive: Line Segments
Primitive: Triangles Tessellated
Primitive types

Instancing

All primitive forms of hair are attached to a mesh that can be instanced to reduce memory consumption.

When line segments are used as the primitive further options include:

- Method - that sets the point in the intersection test at which corrections for the segment width are made.
- Exclude encasing - causes segments, that when scaled by a factor, encase the rays initial location, to be ignored.
- Use tangent normal as default - uses the tangent normal for shading.
- Use tangent normal geometry - uses the tangent normal for the geometry normal.
- Correct tangent normal for slope - corrects the tangent normal for changes in thickness along the segment.
- Ray Mix - allows you to mix the tangent normal with the incoming ray to give a new tangent normal (0 gives normals equal to the incoming ray).

With curve and line primitives there are also options including:

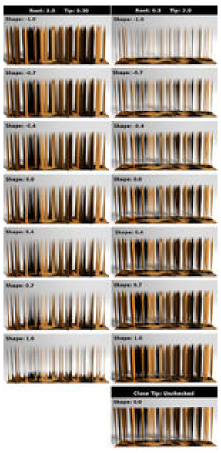
- Check back-faces - sets whether the back face needs including.
- Segments - sets the number of segments inserted between those exported from blender internal.
- Interpolation method - sets the interpolation used to generate the segments.
- Include parents - includes the child strand parents.

Other modes define preset settings for these and are:

- Smooth Curves - uses smooth cardinal curves. This gives the best results but is the slowest.
- Accurate - uses line segments and is suitable for glass shaders.
- True Normal - uses line segments and produces accurate results for opaque shaders.
- Tangent Normal - uses line segments and gives fast results that converge quickly.
- Fast Planes - uses triangles to render planes facing the camera. This uses the most memory but is the fastest to render.

Cycles Hair Settings

The Cycles Hair Settings, under the particle tab, are used to control each hair particle system's strand properties.

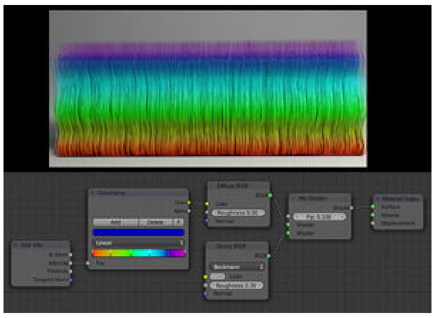

Strand shapes for various settings

These settings include:

- Tip and root multipliers that multiply the particle size to give the Strands thickness. The strand dimensions are also influenced by the object scale.
- A shape parameter that controls the transition in thickness between the root and tip. -1 gives a constant thickness equal to the root, 0 gives a linear transition and 1 gives an instantaneous transition to the tip radius.
- Activating *Close tip* sets the thickness to zero at the tip, even when using a non-zero tip multiplier.

Hair Info Node

The hair info node provides information on the strand for materials.


Example usage of the Hair Info node's intercept output

Limitations

Cycles hair rendering is still being developed and many features are missing. These limitations currently include:

- No motion blur support.
- Only renders in Object mode.
- Scaling along one axis does not always produce the same effect on the hair thickness in the preview and final renders. This can be fixed by applying scale to the mesh.
- The hair rendering primitive settings are global, shared between all particle systems.

Tutorials

- BlenderDiplom Tutorial: - Hair Rendering with Cycles Hair Shaders
- Blender Guru Tutorial - How to render hair with cycles

Light Paths

Ray Types

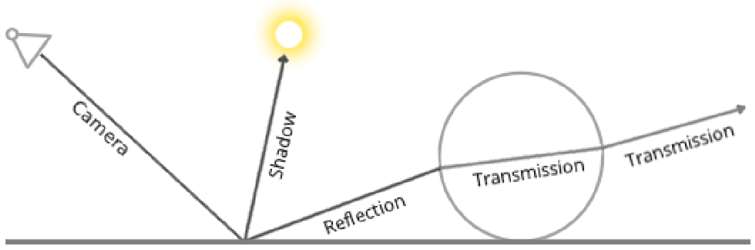
Ray types can be divided into four categories:

- Camera: the ray comes straight from the camera
- Reflection: the ray is generated by a reflection off a surface
- Transmission: the ray is generated by a transmission through a surface
- Shadow: the ray is used for (transparent) shadows

Reflection and transmission rays can further have these properties:

- Diffuse: the ray is generated by a diffuse reflection or transmission (translucency)
- Glossy: the ray is generated by a glossy specular reflection or transmission
- Singular: the ray is generated by a perfectly sharp reflection or transmission

The Light Path node can be used to find out the type of ray the shading is being computed for.



Bounce Control

The maximum number of light bounces can be controlled manually. While ideally this should be infinite, in practice a smaller number of bounces may be sufficient, or some light interactions may be intentionally left out for faster convergence. The number of diffuse reflection, glossy reflection and transmission bounces can also be controlled individually.

Light paths are terminated probabilistically when specifying a minimum number of light bounces lower than the maximum. In that case paths longer than minimum will be randomly stopped when they are expected to contribute less light to the image. This will still converge to the same image, but renders faster while possibly being noisier.

A common source of noise is caustics, which are diffuse bounces followed by a glossy bounce (assuming we start from the camera). An option is available to disable these entirely.

Transparency

The transparent BSDF shader is given special treatment. When a ray passes through it, light passes straight on, as if there was no geometry there. The ray type does not change when passing through a transparent BSDF.

Alpha pass output is also different for the transparent BSDF. Other transmission BSDFs are considered opaque, because they change the light direction. As such they can't be used for alpha-over compositing, while this is possible with the transparent BSDF.

The maximum number of transparent bounces is controlled separately from other bounces. It is also possible to use probabilistic termination of transparent bounces, which might help rendering many layers of transparency.

Note that while semantically the ray passes through as if no geometry was hit, rendering performance is affected as each transparency step requires executing the shader and tracing a ray.

Ray Visibility

Objects can be set to be invisible to particular ray types:

- Camera
- Diffuse reflection
- Glossy reflection
- Transmission
- Shadow
- Volume scatter

This can be used, for example, to make an emitting mesh invisible to camera rays. For duplicators, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

In terms of performance, using these options is more efficient that using a shader node setup that achieves the same effect. Objects invisible to a certain ray will be skipped in ray traversal already, leading to fewer rays cast and shaders executed.

Integrator

The integrator is the rendering algorithm used to compute the lighting. Cycles currently supports a path tracing integrator with direct light sampling. It works well for various lighting setups, but is not as suitable for caustics and some other complex lighting situations.

Rays are traced from the camera into the scene, bouncing around until they find a light source such as a lamp, an object emitting light, or the world background. To find lamps and surfaces emitting light, both indirect light sampling (letting the ray follow the surface BSDF) and direct light sampling (picking a light source and tracing a ray towards it) are used.

Scene Settings

Sampling

There are two integrator modes that can be used: path tracing and branched path tracing. The **path tracing integrator** is a pure path tracer; at each hit it will bounce light in one direction and pick one light to receive lighting from. This makes each individual sample faster to compute, but will typically require more samples to clean up the noise.

Render Samples
: Number of paths to trace for each pixel in the final render. As more samples are taken, the solution becomes less noisy and more accurate.

Preview Samples
: Number of samples for viewport rendering.

The **branched path tracing integrator** (formerly called non-progressive integrator) is similar, but at the first hit it will split the path for different surface components and will take all lights into account for shading instead of just one. This makes each sample slower, but will reduce noise, especially in scenes dominated by direct or one-bounce lighting. To get the same number of diffuse samples as in the path tracing integrator, note that e.g. 250 path tracing samples = 10 AA samples x 25 diffuse samples. The Sampling panel shows this total number of samples.

AA Render Samples
: Number of samples to take for each pixel in the final render. More samples will improve antialiasing.

AA Preview Samples
: Number of samples for viewport rendering.

Diffuse Samples
: Number of diffuse bounce samples to take for each AA sample.

Glossy Samples
: Number of glossy bounce samples to take for each AA sample.

Transmission Samples
: Number of transmission bounce samples to take for each AA sample.

AO Samples
: Number of ambient occlusion samples to take for each AA sample.

Mesh Light Samples
: Number of mesh light samples to take for each AA sample.

Subsurface Samples
: Number of subsurface scattering samples to take for each AA sample.

For both integrators the noise pattern can be controlled.

Seed
: Random number generator seed; each different value gives a different noise pattern.

Bounces

Max Bounces
: Maximum number of light bounces. For best quality, this should be set to the maximum. However, in practice, it may be good to set it to lower values for faster rendering. Setting it to maximum 1 bounce results in direct lighting.

Min Bounces
: Minimum number of light bounces for each path, after which the integrator uses Russian Roulette to terminate paths that contribute less to the image. Setting this higher gives less noise, but may also increase render time considerably. For a low number of bounces, it's strongly recommended to set this equal to the maximum number of bounces.

Diffuse Bounces
: Maximum number of diffuse bounces.

Glossy Bounces
: Maximum number of glossy bounces.

Transmission Bounces
: Maximum number of transmission bounces.

Transparency

Transparency Max
: Maximum number of transparency bounces.

Transparency Min
: Minimum number of transparency bounces, after which Russian Roulette termination is used.

Transparent Shadows
: For direct light sampling, use transparency of surfaces in between to produce shadows affected by transparency of those surfaces.

Tricks

No Caustics
: While in principle path tracing supports rendering of caustics with a sufficient number of samples, in practice it may be inefficient to the point that there is just too much noise. This option makes it possible to disable them entirely.

Filter Glossy
: When using a value higher than 0.0, this will blur glossy reflections after blurry bounces, to reduce noise at the cost of accuracy. 1.0 is a good starting value to tweak.

Some light paths have a low probability of being found while contributing much light to the pixel. As a result these light paths will be found in some pixels and not in others, causing fireflies. An example of such a difficult path might be a small light that is causing a small specular highlight on a sharp glossy material, which we are seeing through a rough glossy material. In fact in such a case we practically have a caustic.

With path tracing it is difficult to find the specular highlight, but if we increase the roughness on the material, the highlight gets bigger and softer, and so easier to find. Often this blurring will hardly be noticeable, because we are seeing it through a blurry material anyway, but there are also cases where this will lead to a loss of detail in lighting.

Clamp Samples
: This option will clamp all samples to a maximum intensity they can contribute to the pixel, again to reduce noise at the cost of accuracy. With value 0.0 this option is disabled; lower values clamp more light away.

If the image has fireflies, there will be samples that contribute very high values to pixels, and this option provides a way to limit that. However note that as you clamp out such values, bright colors in other places where there is no noise will be lost as well. So this is a balance between reducing the noise and keeping the image from losing its intended bright colors.

Motion Blur

Camera and object motion blur rendering can be enabled per scene, and affects all render layers. This will take the camera and object motion into account to blur objects along 3 points through the previous, current and next frame. Currently scale motion is not supported, only object transformations like translation and rotation. Viewport rendering currently will not show motion blur.

If there are particles or other physics system in a scene, be sure to bake them before rendering, otherwise you might not get correct or consistent motion.

Shutter
: Time between frames over which motion blur is computed. Shutter time 1.0 blurs over the length of 1 frame, 2.0 over the length of two frames, from the previous to the next.

Material Settings

Multiple Importance Sample
: By default objects with emitting materials use both direct and indirect light sampling methods, but in some cases it may lead to less noise overall to disable direct light sampling for some materials. This can be done by disabling the Multiple Importance Sample option. This is especially useful on large objects that emit little light compared to other light sources.

This option will only have an influence if the material contains an emission node; it will be automatically disabled otherwise.

World Settings

Multiple Importance Sample
: By default lighting from the world is computed solely with indirect light sampling. However for more complex environment maps this can be too noisy, as sampling the BSDF may not easily find the highlights in the environment map image. By enabling this option, the world background will be sampled as a lamp, with lighter parts automatically given more samples.

Map Resolution
: When Multiple Importance Sample is enabled, this specifies the size of the importance map (resolution x resolution). Before rendering starts, an importance map is generated by "baking" a grayscale image from the world shader. This will then be used to determine which parts of the background are light and so should receive more samples than darker parts. Higher resolutions will result in more accurate sampling but take more setup time and memory.

Lamp Settings

Multiple Importance Sample
: By default lamps use only direct light sampling. For area lights and sharp glossy reflections, however, this can be noisy, and enabling this option will enable indirect light sampling to be used in addition to reduce noise.

Samples
: For the branch path tracing integrator, this specifies the number of direct light samples per AA sample. Point lamps might need only one sample, while area lamps typically need more.

Volume Render Settings

The scene has these settings:

Step Size

Distance between volume shader samples when rendering the volume. Lower values give more accurate and detailed results but also increased render time.

Max Steps
: Maximum number of steps through the volume before giving up, to protect from extremely long render times with big objects or small step sizes.

The world and materials have the following setting:

Homogeneous Volume
: Assume volume has the same density everywhere (not using any textures), for faster rendering. For example absorption in a glass object would typically not have any textures, and by knowing this we can avoid taking small steps to sample the volume shader.

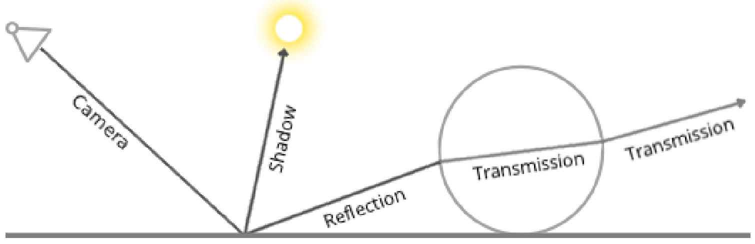
Sampling Method
: Options are "Multiple Importance", "Distance" or "Equiangular". If you've got a pretty dense volume that's lit from far away then distance sampling is usually more efficient. If you've got a light inside or near the volume then equiangular sampling is better. If you have a combination of both, then the multiple importance sampling will be better.

Reducing Noise

When performing a final render, it is important to reduce noise as much as possible. Here we'll discuss a number of tricks that, while breaking the laws of physics, are particularly important when rendering animations within a reasonable time. Click to enlarge the example images to see the noise differences well.

Path Tracing

Cycles uses path tracing with next event estimation, which is not good at rendering all types of light effects, like caustics, but has the advantage of being able to render more detailed and larger scenes compared to some other rendering algorithms. This is because we do not need to store, for example, a photon map in memory, and because we can keep rays relatively coherent to use an on-demand image cache, compared to e.g. bidirectional path tracing.

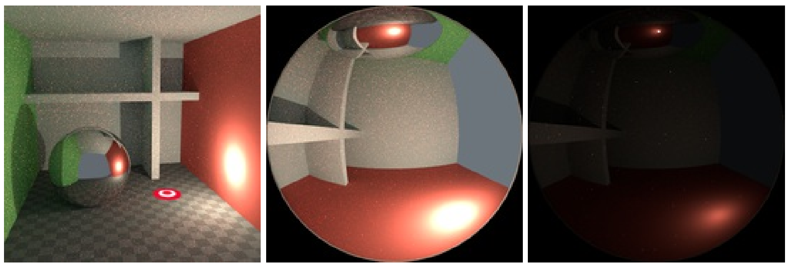


We do the inverse of what reality does, tracing light rays from the camera into the scene and onto lights, rather than from the light sources into the scene and then into the camera. This has the advantage that we do not waste light rays that will not end up in the camera, but also means that it is difficult to find some light paths that may contribute a lot. Light rays will be sent either according to the surface BRDF, or in the direction of known light sources (lamps, emitting meshes with Sample as Lamp).

For more details, see the Light Paths and Integrator documentation.

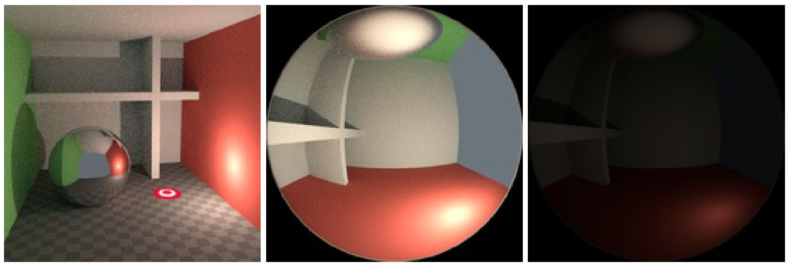
Where Noise Comes From

To understand where noise can come from, take for example this scene. When we trace a light ray into the specified location, this is what the diffuse shader "sees". To find the light that is reflected from this surface, we need to find the average color from all these pixels. Note the glossy highlight on the sphere, and the bright spot the lamp casts on the nearby wall. These hotspots are 100x brighter than other parts of the image and will contribute significantly to the lighting of this pixel.



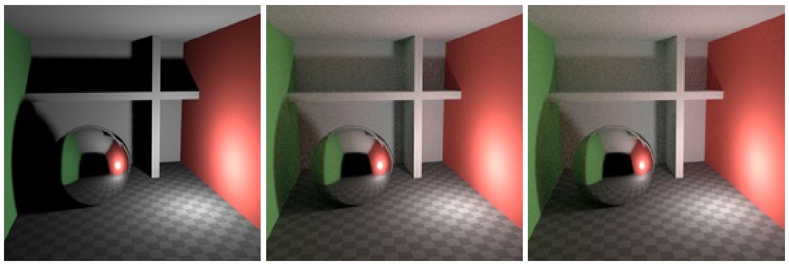
The lamp is a known light source, so it will not be too hard to find, but the glossy highlight(s) that it causes are a different matter. The best we can do with path tracing is to distribute light rays randomly over the hemisphere, hoping to find all the important bright spots. If for some pixels we miss some bright spot, but we do find it for another, that results in noise. The more samples we take, the higher the probability that we cover all the important sources of light.

With some tricks we can reduce this noise. If we blur the bright spots, they become bigger and less intense, making them easier to find and less noisy. This will not give the same exact result, but often it's close enough when viewed through a diffuse or soft glossy reflection. Below is an example of using Filter Glossy and Smooth Light Falloff.



Bounces

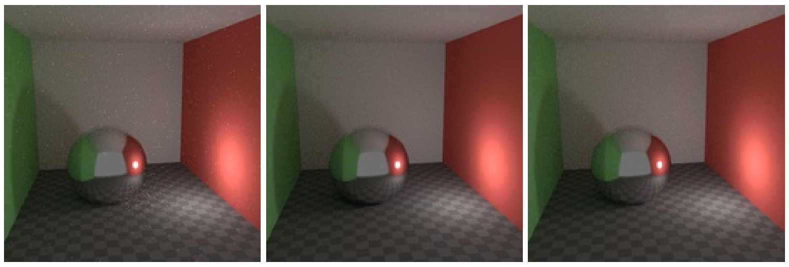
In reality light will bounce a huge number of times due to the speed of light being very high. In practice more bounces will introduce more noise, and it might be good to use something like the Limited Global Illumination preset that uses **fewer bounces for different shader types**. Diffuse surfaces typically can get away with fewer bounces, while glossy surfaces need a few more, and transmission shaders such as glass usually need the most.



Also important is to **use shader colors that do not have components of value 1.0** or values near that; try to keep the maximum value to 0.8 or less and make your lights brighter. In reality, surfaces are rarely perfectly reflecting all light, but there are of course exceptions; usually glass will let most light through, which is why we need more bounces there. High values for the color components tend to introduce noise because light intensity then does not decrease much as it bounces off each surface.

Caustics and Filter Glossy

Caustics are a well-known source of noise, causing fireflies. They happen because the renderer has difficulty finding specular highlights viewed through a soft glossy or diffuse reflection. There is a No Caustics option to disable glossy behind a diffuse reflection entirely. Many render engines will typically disable caustics by default.

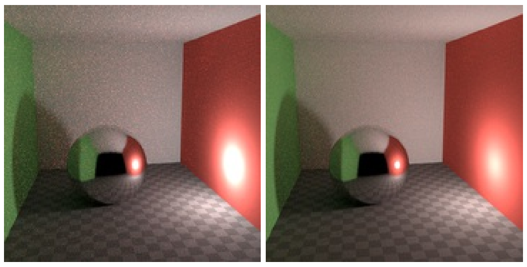


However using No Caustics will result in missing light, and it still does not cover the case where a sharp glossy reflection is viewed through a soft glossy reflection. There exists a Filter Glossy option to reduce the noise from such cases at the cost of accuracy. This will blur the sharp glossy reflection to make it easier to find, by increasing the shader Roughness.

The above images show default settings, no caustics, and filter glossy set to 1.0.

Light Falloff

In reality light in a vacuum will always fall off at a rate of 1/(distance^2). However as distance goes to zero, this value goes to infinity and we can get very bright spots in the image. These are mostly a problem for indirect lighting, where the probability of hitting such a small but extremely bright spot is low and so happens only rarely. This is a typical recipe for fireflies.

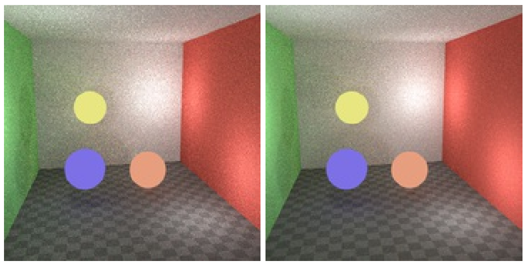


To reduce this problem, the Light Falloff node has a **Smooth factor, that can be used to reduce the maximum intensity** a light can contribute to nearby surfaces. The images above show default falloff and smooth value 1.0.

Sample as Lamp

Materials with emission shaders can be configured to be Sampled as a Lamp. This means that they will get rays sent directly towards them, rather than ending up there based on rays randomly bouncing around. For very bright mesh light sources, this can reduce noise significantly. However when the emission is not particularly bright, this will take samples away from other brighter light sources for which it is important to find them this way.

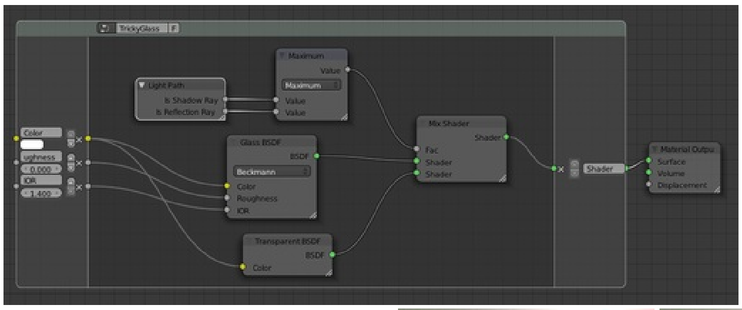
The optimal setting here is difficult to guess; it may be a matter of trial and error, but often it is clear that a somewhat glowing object may be only contributing light locally, while a mesh light used as a lamp would need this option enabled. Here is an example where the emissive spheres contribute little to the lighting, and the image renders with slightly less noise by disabling Sample as Lamp on them.

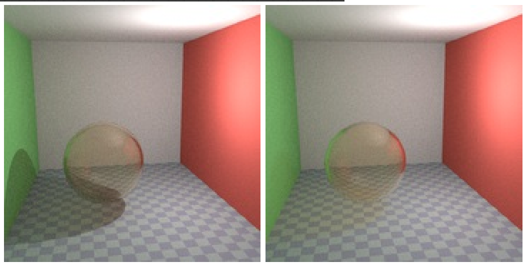


The world background also has a Sample as Lamp option. This is mostly useful for environment maps that have small bright spots in them, rather than being smooth. This option will then, in a preprocess, determine the bright spots, and send light rays directly towards them. Again, enabling this option may take samples away from more important light sources if it is not needed.

Glass and Transparent Shadows

With caustics disabled, glass will miss shadows, and with filter glossy they might be too soft. We can make a glass shader that will **use a Glass BSDF when viewed directly, and a Transparent BSDF when viewed indirectly**. The Transparent BSDF can be used for transparent shadows to find light sources straight through surfaces, and will give properly-colored shadows, but without the caustics. The Light Path node is used to determine when to use which of the two shaders.



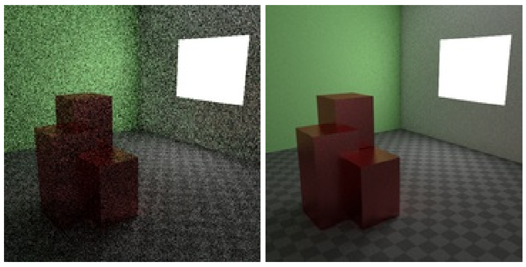


Above we can see the node setup used for the glass transparency trick; on the left the render has too much shadow due to missing caustics, and on the right the render with the trick.

Window Lights

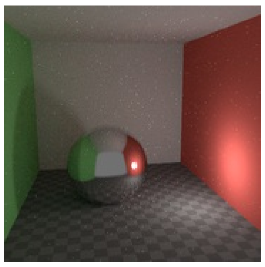
When rendering a daylight indoor scene where most of the light is coming in through a window or door opening, it is difficult for the integrator to find its way to them. We can replace the opening with a plane with an emission shader, so that the integrator knows in which direction to fire rays. For camera rays we can make this mesh light invisible, so that we can still look into the outside scene. This is done either by disabling camera ray visibility on the object, or by switching between glass and emission shaders in the material.

The two renders below have the same render time, with the second render using a mesh light positioned in the window.



Clamp Fireflies

Ideally with all the previous tricks, fireflies would be eliminated, but they could still happen. For that, **the intensity that any individual light ray sample will contribute to a pixel can be clamped** to a maximum value with the integrator Clamp setting. If set too low this can cause missing highlights in the image, which might be useful to preserve for camera effects such as bloom or glare.



Render Passes

Layers

Render layers are used to render different objects in the scene into different images. This way they can, for example, be color corrected or otherwise manipulated separately and then recomposed in compositing later.

Which objects contribute to which render layers are defined by these layer settings:

- Scene Layers: only objects on these layers will contribute to the image.
- Camera Layers: objects on these layers are directly visible to the camera. When an object is in the scene layers but not camera layers, it will still cast shadows or be visible in reflections, so it's still indirectly visible. This is equivalent to disabling the Camera in the Ray Visibility panel for the object. The way this works may be somewhat confusing at first, but it's designed such that render layers can be recomposed to give the full render, without any missing shadows or reflections.
- Mask Layers: objects on these will mask out other objects appearing behind them. This is equivalent to assigning a Holdout shader for camera rays to the objects on such layers.
- Exclude Layers: scene layers are shared between all render layers; however sometimes it's useful to leave out some object influence for a particular render layer. That's what this option allows you to do.

Lighting Passes

Diffuse Direct
: Direct lighting from diffuse BSDFs. We define direct lighting as coming from lamps, emitting surfaces, the background, or ambient occlusion after a single reflection or transmission off a surface. BSDF color is not included in this pass.

Diffuse Indirect
: Indirect lighting from diffuse BSDFs. We define indirect lighting as coming from lamps, emitting surfaces or the background after more than one reflection or transmission off a surface. BSDF color is not included in this pass.

Diffuse Color
: Color weights of diffuse BSDFs. These weights are the color input socket for BSDF nodes, modified by any Mix and Add Shader nodes.

Glossy Direct, Indirect, Color
: Same as above, but for glossy BSDFs.

Transmission Direct, Indirect, Color
: Same as above, but for transmission BSDFs.

Subsurface Direct, Indirect, Color
: Same as above, but for subsurface BSDFs.

Emission
: Emission from directly visible surfaces.

Environment
: Emission from the directly visible background. When the film is set to transparent, this can be used to get the environment color and composite it back in.

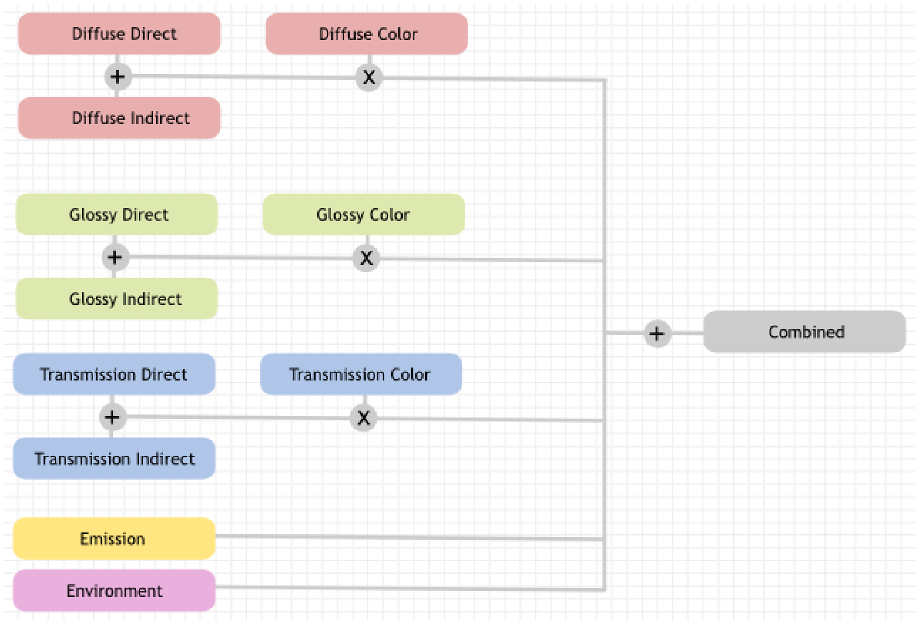
Shadow
: Shadows from lamp objects.

Ambient Occlusion
: Ambient occlusion from directly visible surfaces. BSDF color or AO factor is not included; i.e. it gives a 'normalized' value between 0 and 1.

Note that transparent BSDFs are given special treatment: a fully transparent surface is treated as if there is no surface there at all; a partially transparent surface is treated as if only part of the light rays can pass through. This means it is not included in the Transmission passes; for that a glass BSDF with index of refraction 1.0 can be used.

Combining

All these lighting passes can be combined to produce the final image as follows:



Data Passes

Z
: Z depth.

Mist
: Mist value between 0.0 and 1.0, using settings from the Mist Pass panel in world properties.

Normal
: Surface normal used for shading.

UV
: Default render UV coordinates.

Object Index
: Pass index of object.

Material Index
: Pass index of material.

Vector
: Motion vectors for the vector blur node. The four components consist of 2D vectors giving the motion towards the next and previous frame position in pixel space.

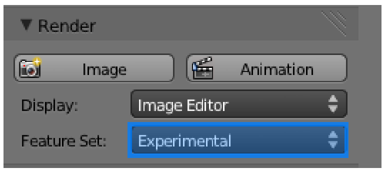
The Z, Object Index and Material Index passes are not antialiased. This is done because such values can't really be blended correctly.

Alpha Threshold
: Z, Index, normal, UV and vector passes are only affected by surfaces with alpha transparency equal to or higher than this threshold. With value 0.0 the first surface hit will always write to these passes, regardless of transparency. With higher values surfaces that are mostly transparent can be skipped until an opaque surface is encountered.

Experimental Features

Some features in Cycles are not finished yet, but already included in builds for testing. These features may not work, crash Blender or change behaviour in later versions.

They are hidden by default, but can be enabled by setting Feature Set to Experimental in the Render properties.



Display: Image Editor
Feature Set: Experimental

Currently considered experimental:

- OpenCL device
- Displacement
- Subdivision
- GPU: Subsurface Scattering and Correlated Multi Jitter

GPU Rendering

Introduction

GPU rendering makes it possible to use your graphics card for rendering, instead of the CPU. This can speed up rendering, because modern GPUs are designed to do quite a lot of number crunching. On the other hand, they also have some limitations in rendering complex scenes, due to more limited memory, and issues with interactivity when using the same graphics card for display and rendering.

Cycles has two GPU rendering modes: through CUDA, which is the preferred method for NVidia graphics cards; and OpenCL, which is intended to support rendering on AMD graphics cards. The implementation of OpenCL is only in an experimental stage and disabled in official builds.

Configuration

To enable GPU rendering, go into the User Preferences, and under the System tab, select the Compute Device(s) to use. Next, for each scene, you can configure to use CPU or GPU rendering in the Render properties.

CUDA

NVidia CUDA is supported for GPU rendering with **NVidia graphics cards**. We support graphics cards starting from GTX 4xx (computing capability 2.0).

Cycles requires recent drivers to be installed, on all operating systems. Be sure to download the Blender version matching your operating system; that is, download 64-bit Blender for 64-bit operating systems.

List of CUDA cards with shader model

Limitations

- The maximum amount of individual textures is limited to 95 byte-image textures (PNG, JPEG, ..) and 5 float-image textures (OpenEXR, 16 bit TIFF, ..) on GTX 4xx/5xx cards, and 145 byte-image textures and 5 float-image textures on GTX6xx cards and above.
- Open shading language (OSL) is only supported by CPU.
- Smoke/Fire rendering is not supported on GPU.

Frequently Asked Questions

Why is Blender unresponsive during rendering?

While a graphics card is rendering, it can not redraw the user interface, which makes Blender unresponsive. We attempt to avoid this problem by giving back control over the GPU as often as possible, but a completely smooth interaction can't be guaranteed, especially on heavy scenes. This is a limitation of graphics cards for which no true solution exists, though we might be able to improve this somewhat in the future.

If possible, it is best to install more than one GPU, using one for display and the other(s) for rendering.

Why does a scene that renders on the CPU not render on the GPU?

There maybe be multiple causes, but the most common is that there is not enough memory on your graphics card. We can currently only render scenes that fit in graphics card memory, and this is usually smaller than that of the CPU. Note that, for example, 8k, 4k, 2k and 1k image textures take up respectively 256MB, 64MB, 16MB and 4MB of memory.

We do intend to add a system to support scenes bigger than GPU memory, but this will not be added soon.

Can I use multiple GPUs for rendering?

Yes, go to User Preferences > System > Compute Device Panel, and configure it as you desire.

Would multiple GPUs increase available memory?

No, each GPU can only access its own memory.

What renders faster, NVidia or AMD, CUDA or OpenCL?

Currently NVidia with CUDA is rendering faster. There is no fundamental reason why this should be so—we don't use any CUDA-specific features—but the compiler appears to be more mature, and can better support big kernels. OpenCL support is still being worked on and has not been optimized as much, because we haven't had the full kernel working yet.

Error Messages

Unsupported GNU version! gcc 4.7 and up are not supported!

On Linux, depending on your GCC version you might get this error.

If so, delete the following line in /usr/local/cuda/include/host_config.h

```
#error -- unsupported GNU version! gcc 4.7 and up are not supported!
```

CUDA Error: Invalid kernel image

If you get this error on Windows 64-bit, be sure to use the 64-bit build of Blender, not the 32-bit version.

CUDA Error: Out of memory

This usually means there is not enough memory to store the scene on the GPU. We can currently only render scenes that fit in graphics card memory, and this is usually smaller than that of the CPU. See above for more details.

The NVIDIA OpenGL driver lost connection with the display driver

... due to exceeding the Windows Time-Out limit and is unable to continue.

If a GPU is used for both display and rendering, Windows has a limit on the time the GPU can do render computations. If you have a particularly heavy scene, Cycles can take up too much GPU time. Reducing Tile Size in the Performance panel may alleviate the issue, but the only real solution is to use separate graphics cards for display and rendering.

Another solution can be to increase the timeout, although this will make the user interface less responsive when rendering heavy scenes. http://msdn.microsoft.com/en-us/windows/hardware/gg487368.aspx

CUDA error: Unknown error in cuCtxSynchronize()

An unknown error can have many causes, but one possibility is that it's a timeout. See the above answer for solutions.

On Mac OS X, no CUDA GPU is available

Make sure you have the CUDA driver (any recent version) installed.

If it still doesn't work, ensure that in the Energy Saver preferences, the automatic graphics switching is disabled and the fastest GPU is selected. Also ensure you do not have other CUDA toolkit versions installed.

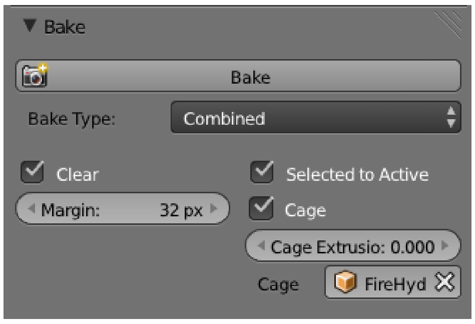
Render Baking

Refer to Blender main Render page for general baking guidelines: http://wiki.blender.org/index.php/Doc:2.6/Manual/Render/Bake

Cycles uses the render settings (samples, bounces, ...) for baking. This way the quality of the baked textures should match the result you get from the rendered scene.

The baking happens into the respective active textures of the object materials. The active texture is the last selected Image Texture node of the material nodetree. That means the active object (or the selected objects, when not baking 'Selected to Active') needs a material, and that material needs at least an Image Texture node, with the image to be used for the baking. Note, the node doesn't need to be connected to any other node. The active texture is what projection painting and the viewport use as a criteria to which image to use. This way after the baking is done you can automatically preview the baked result in the Texture mode.

Options



Combined Pass

Bake Mode

Combined

Bakes all materials, textures, and lighting except specularity and SSS.

Ambient Occlusion

Bakes ambient occlusion as specified in the World panels. Ignores all lights in the scene.

Shadow

Bakes shadows and lighting.

Normals

Bakes normals to an RGB image.

Normal Space
: Normals can be baked in different spaces:

World space

Normals in world coordinates, dependent on object transformation and deformation.

Object space

Normals in object coordinates, independent of object transformation, but dependent on deformation.

Tangent space

Normals in tangent space coordinates, independent of object transformation and deformation. This is the default, and the right choice in most cases, since then the normal map can be used for animated objects too.

Normal Swizzle

Axis to bake into the red, green and blue channel.

For materials the same spaces can be chosen as well, in the image texture options, next to the existing Normal Map setting. For correct results, the setting here should match the setting used for baking.

UV

Bakes colors of materials and textures only, without shading.

Emit

Bakes Emission, or the Glow color of a material.

Environment

Bakes the environment as seen from the center of the object.

Diffuse Color/Direct/Indirect

Bakes the diffuse pass of a material. Diffuse Color is a property of the surface and independent of sampling refinement.

Glossy Color/Direct/Indirect

Bakes the glossiness of a material. Glossy Color is a property of the surface and independent of sampling refinement.

Transmission Color/Direct/Indirect

Bakes the transmission of a material. Transmission Color is a property of the surface and independent of sampling refinement.

Subsurface Color/Direct/Indirect

Bakes the subsurface pass of a material. Subsurface Color is a property of the surface and independent of sampling refinement.

Additional Options

Margin
: Baked result is extended this many pixels beyond the border of each UV "island," to soften seams in the texture.

Clear
: If selected, clears the image before baking render.

Select to Active
: Bake shading on the surface of selected objects to the active object. The rays are cast from the lowpoly object inwards towards the highpoly object. If the highpoly object is not entirely involved by the lowpoly object, you can tweak the rays start point with **Ray Distance** or **Cage Extrusion** (depending on whether or not you are using cage). For even more control you can use a **Cage Object**.

Memory Usage
There is a CPU fixed memory footprint for every object used to bake from. In order to avoid crashes due to lack of memory the highpoly objects can be joined before the baking process. The render tiles parameter also influence the memory usage, so the bigger the tile the less overhead you have, but the more memory it will take during baking (either in GPU or CPU).

Cage
: Cast rays to active object from a cage. A cage is a ballooned-out version of the lowpoly mesh created either automatically (by adjusting the ray distance) or manually (by specifying an object to use). When not using a cage the rays will conform to the mesh normals. This produces glitches on the edges, but it's a preferable method when baking into planes to avoid the need of adding extra loops around the edges.

Ray Distance
: Distance to use for the inward ray cast when using selected to active. Ray distance is only available when not using **Cage**.

Cage Extrusion
: Distance to use for the inward ray cast when using **Selected to Active** and **Cage**. The inward rays are casted from a version of the active object with disabled Edge Split modifiers. Hard splits (e.g., when the Edge Split modifier is applied) should be avoided because they will lead to non-smooth normals around the edges.

Cage
: Object to use as cage instead of calculating the cage from the active object with the **Cage Extrusion**.

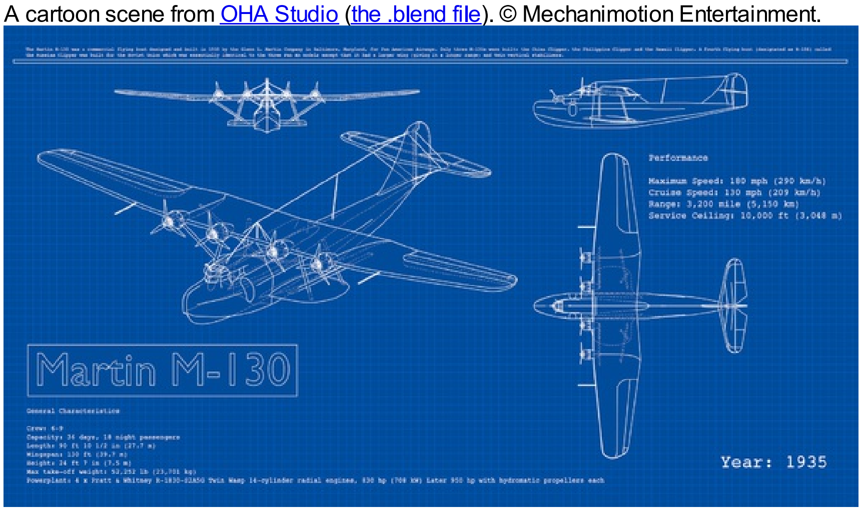
Cage
When the base mesh extruded doesn't give good results, you can create a copy of the base mesh, and modify it to use as Cage. Both meshes need to have the same topology (number of faces and face order).

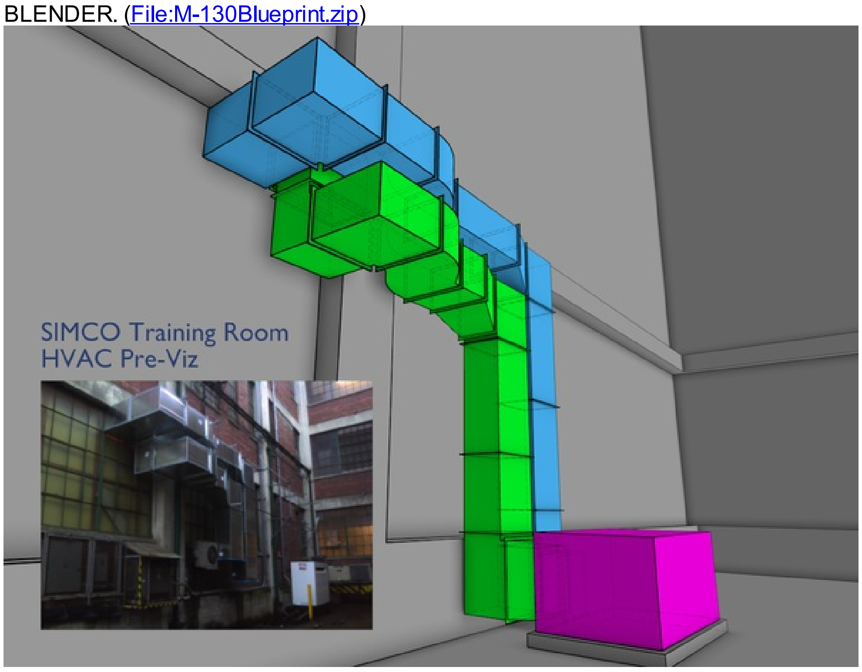
What is FreeStyle?

Freestyle is an edge- and line-based non-photorealistic (NPR) rendering engine. It relies on mesh data and z-depth information to draw lines on selected edge types. Various line styles can be added to produce artistic (“hand drawn”, “painted”, etc.) or technical (hard line) looks.

The two operating modes - Python Scripting and Parameter Editor - allow a powerful diversity of line styles and results. Line styles such as Japanese big brush, cartoon, blueprint, thickness-with-depth are already pre-scripted in Python. The Parameter Editor mode allows intuitive editing of features such as dotted lines and easy setup of multiple line types and edge definitions. On top of all of that, with the introduction of line style modifiers, the sky is the limit!

A cartoon scene from OHA Studio (the .blend file). © Mechanimotion Entertainment.


Martin M-130
Performance
Maximum Speed: 180 mph (290 km/h)
Cruise Speed: 130 mph (209 km/h)
Range: 3,200 mile (5,150 km)
Service Ceiling: 10,000 ft (3,048 m)
General Characteristics
Year: 1935
Blueprint render of Martin M-130 from 1935 by LightBWK. CC0. WARNING: HEAVY FILE! DESIGNED FOR STRESS TEST BLENDER TO THE LIMITS & MAY CRASH BLENDER. (File:M-130Blueprint.zip)



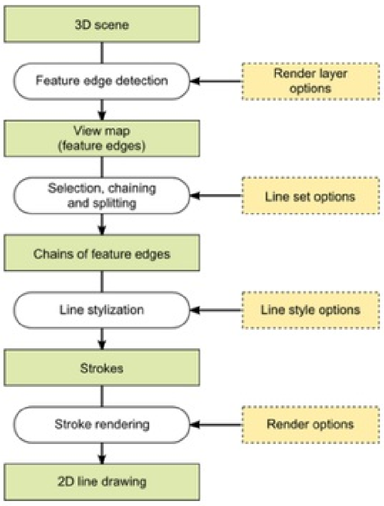
HVAC Pre-Viz by Lee Posey. CC0 (File:HVACPreViz.zip)

Kitchen by Vicente Carro. © AnigoAnimation

More artwork can be found at http://wiki.blender.org/index.php/Dev:Ref/Release_Notes/2.67/FreeStyle#Freestyle_Artwork_Showcase

The Big Picture

1. Activate FreeStyle by Properties window → Render tab → FreeStyle panel, tick check box. Please note that FreeStyle is only available for the Blender Internal renderer.
2. Freestyle settings are located in the new Render Layers context.
3. One render layer can only have one viewmap. A viewmap holds the edge detection settings (Crease Angle, Culling toggle, Face Smoothness toggle, Material Boundaries toggle, Sphere Radius and Kr Derivative Epsilon advanced options).
4. A viewmap can have multiple line sets.
5. A line set controls which line types and selections will be rendered, from lines based on your scene.
6. Each line set uses one line style (which can be shared between multiple line sets).
7. A line style tells Freestyle how to render the linked line sets in terms of color, alpha, thickness and other aspects.

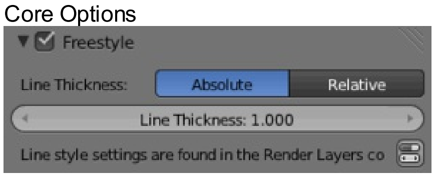


block diagram of Freestyle view map and processes

Known Limitations and issues

- FreeStyle is only available for the Blender Internal renderer.
- Highly memory demanding: All mesh objects in a render layer are loaded at once.
- Only faced mesh objects are supported. The following kinds of meshes are ignored.
  - Mesh faces with wire materials.
  - Mesh faces with completely transparent materials.
- Transparent faces are treated as opaque faces.
- No edges at face intersections are detected yet.
- Layer masks do not work with Freestyle.
- Freestyle rendering results do not have any Z depth information.
- Does not work with a panoramic camera.

Core Options



Freestyle core options.

Activating Freestyle in the Render context of the Buttons window will give you the following options:

Line Thickness
: There are two different modes for defining the base line thickness:

  Absolute

  : The line thickness is given by a user-specified number of pixels. The default value is **1.0**.

  Relative

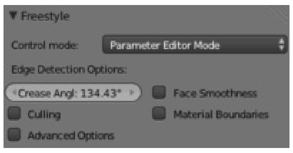
  : The unit line thickness is scaled by the proportion of the present vertical image resolution to **480** pixels. For instance, the unit line thickness is **1.0** with the image height set to **480**, **1.5** with **720**, and **2.0** with **960**.

Line Thickness
: Only for Absolute line thickness: base line thickness in pixels, **1.0** by default.

Viewmaps

There is only one viewmap per render layer. It controls the edge detection parameters. Which detected edges are actually rendered, and how, can be controlled either through the user-friendly parameter editor, or powerful but complex Python scripting.

Face Smoothness
: When enabled, Face Smoothness will be taken into account for edges calculation.



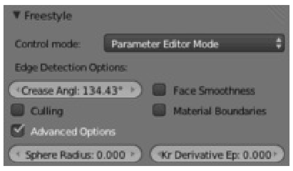
▼ Freestyle
Control mode: Parameter Editor Mode
Edge Detection Options:
Crease Angl: 134.43°
Face Smoothness
Culling
Material Boundaries
Advanced Options

Parameter Editor Mode UI

Crease Angle
: If two adjacent faces form an angle less than the defined Crease Angle, the edge between them will be rendered when using Crease edge type selection in a line set. The value also affects Silhouette edge type selection.

Culling
: Ignore the edges that are out of view (saves some processing time and memory, but may reduce the quality of the result in some cases).

Advanced Options



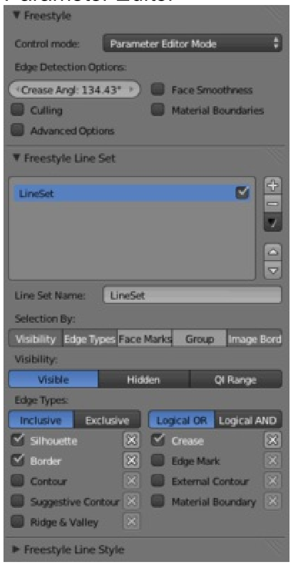
▼ Freestyle
Control mode: Parameter Editor Mode
Edge Detection Options:
Crease Angl: 134.43°
Face Smoothness
Culling
Material Boundaries
✓ Advanced Options
Sphere Radius: 0.000
Kr Derivative Ep: 0.000

Advanced Options

Sphere Radius
It affects the calculation of curvatures for Ridge, Valley and Suggestive Contour edge type selection in a line set.

Kr Derivative Epsilon
: It provides you with control over the output of Suggestive Contour and Silhouette edge type selection (further information in File:Manual-2.6-Render-Freestyle-PrincetownLinestyle.pdf).

Parameter Editor



Control mode: Parameter Editor Mode

Crease Angl: 134.43°

Line Set Name: LineSet

Parameter Editor

The Freestyle Parameter Editor is a user-friendly interface to define and control line sets and line styles.

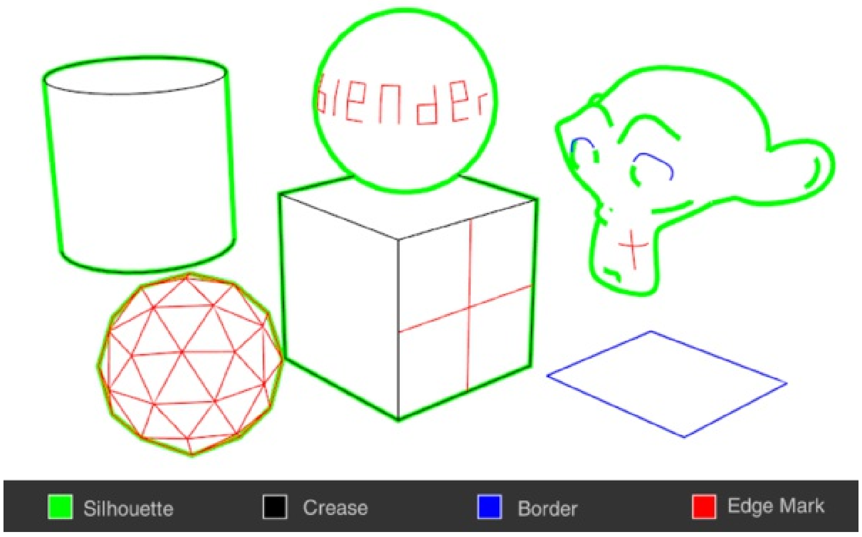
Line sets control which of the edges detected by Freestyle will actually be used (rendered).

Line styles control how the selected edges are rendered.

A view map (hence a render layer) can have multiple line sets, and each line set is linked to one line style.

Line Set

A line set selects, among the lines (edges) detected by Freestyle, which ones will be rendered using its attached line style, through various methods.



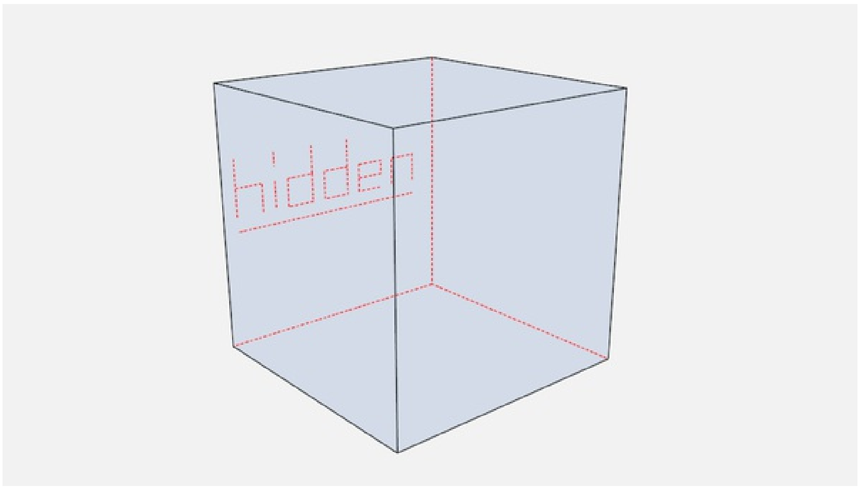
Examples of some basic edge types by LightBWK (File:EdgeType.zip)

Selection by Visibility

There are three choices for selecting edges by visibility.

Visible
: Only lines occluded by no surfaces are rendered.

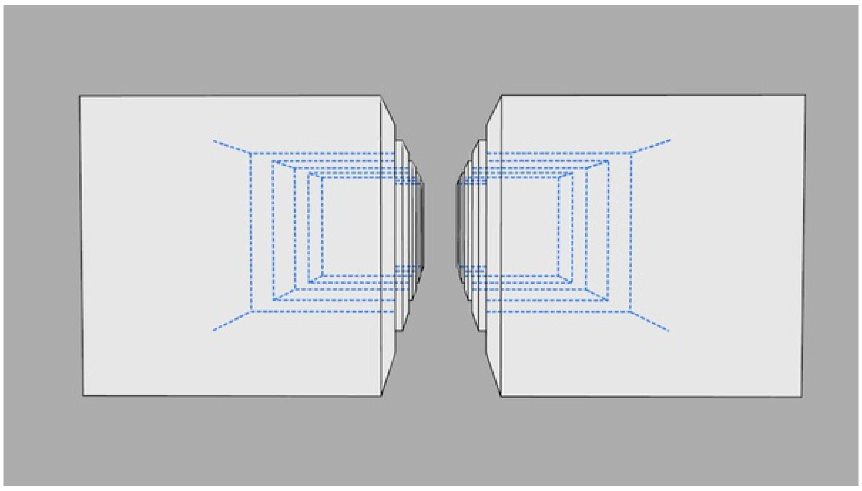
Hidden
: Lines occluded by at least one surface are rendered.



Proof of concept of visible and hidden edges by LightBWK (File:HiddenCreaseEdgeMark.zip)

QI Range
: QI stands for *Quantitative Invisibility*. Lines occluded by a number of surfaces in the given range are rendered.

Start and End
: Only with QI Range, min/max number of occluding surfaces for a line to be rendered.

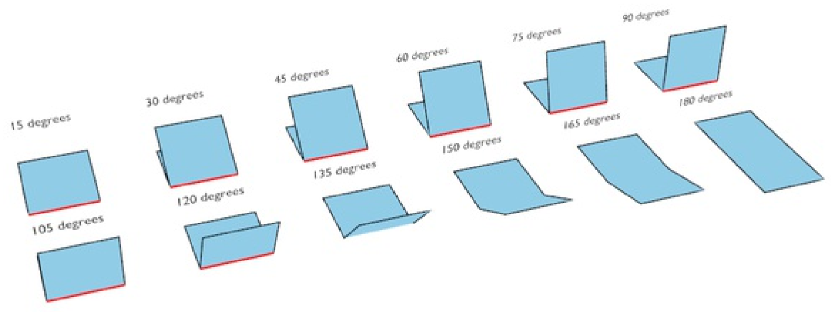

QI Range proof of concept demo, Start: 3, End: 7, by LightBWK (File:QI-Range.zip)

Selection by Edge Types

Edge types are basic algorithms for the selection of lines from geometry. When using the parameter editor you have to choose at least one edge type in order to get a render output, but several edge types can be combined in one line set. Edge types can also be excluded from calculation by pressing the X next to them.

Silhouette
: Draws silhouettes around your closed objects; it is often good for organic objects (like Suzanne & Sphere), and bad for sharp edges, like a box. It can’t render open mesh objects like open cylinders and flat planes. The output is affected by the Kr Derivative Epsilon viewmap setting.

Crease
: Shows only edges whose adjacent faces form an angle greater than the defined viewmap’s Crease Angle.



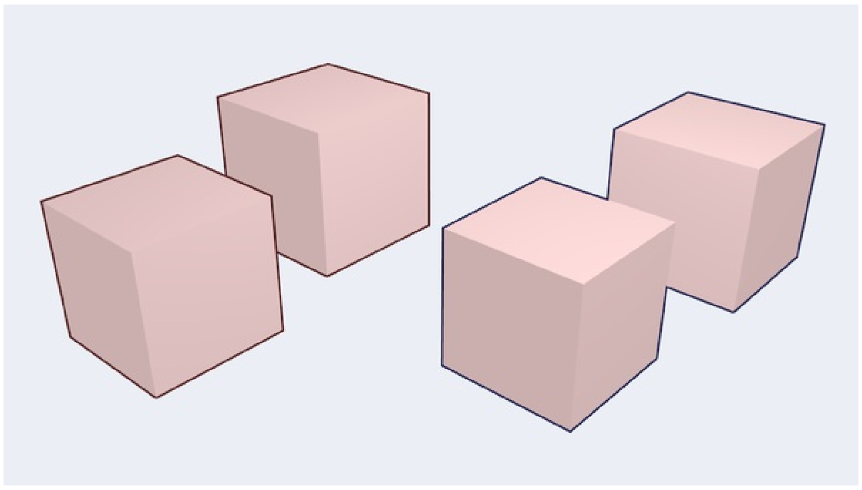
Crease Angle proof of concept for 121° by LightBWK (File:CreaseAngle.zip)

Border
: Border is for open/unclosed edge meshes; an open cylinder has an open edge at the top and bottom, and a plane is open all around. Suzanne's eye socket is an open edge. All open edges will have lines rendered. This depends on the mesh structure.

Edge Marks
: Renders marked edges. See below for details.

Contour
: Draws the outer edges and inner open border.

External Contour
: Draws the contour lines, but only on the outer edges.

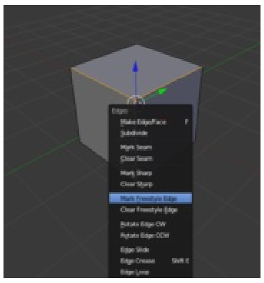

Left pair: Contour; Right pair: External Contour

Suggestive Contour
: Draws some lines which would form the contour of the mesh if the viewport was shifted. Depends on your viewmap settings for Kr Derivative Epsilon and Sphere Radius (further information: File:Manual-2.6-Render-Freestyle-PrincetownLinestyle.pdf).

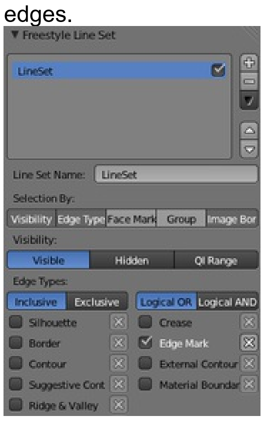
Material Boundary
: Draws lines where two materials meet on the same object. Must be activated in the viewmap settings.

Ridge & Valley
: Draws ridges and valleys. Depends on your Sphere Radius viewmap settings.

Edge Marks



Select and mark Freestyle edges.


▼ Freestyle Line Set

LineSet ✓

Line Set Name: LineSet

Selection By: Visibility | Edge Type | Face Mark | Group | Image Bor

Visibility: [✓ Visible] | Hidden | QI Range

Edge Types: [✓ Inclusive] | Exclusive   [✓ Logical OR] | Logical AND

Silhouette	Crease
Border	✓ Edge Mark
Contour	External Contour
Suggestive Cont	Material Boundar
Ridge & Valley	

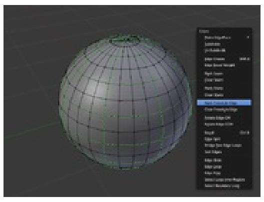
Edge Mark setting in the Line Sets tab.

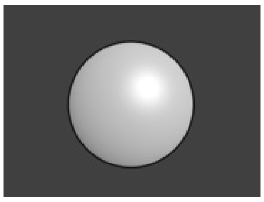
In edit mode you can mark “Freestyle Edges” in the same manner you can mark “Seams” for UV unwrapping or “Sharp” for edge split. These marked edges are available to render when you select Edge Mark.

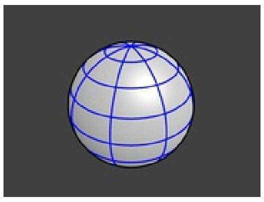
This is done as follows:

- Select your mesh and tab into Edit mode.
- Select the edges you want to be marked.
- Press CrtlE and select Mark Freestyle Edge.

Edge marks are useful when you want to draw lines along particular mesh edges. The examples below explain the use of edge marks.


Marking Freestyle Edges in edit mode.


Render without Edge Marks.


Render with Edge Marks enabled.

The image on the left shows a sphere in Edit mode. The green lines are the edge marks. On the right you see a render without edge marks enabled.

With edge marks enabled, the previously-marked lines are always rendered. You can see the black contour lines and the blue lines that are made with edge marks.

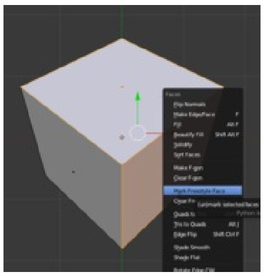
What are edge marks good for?

1. When you need to render marks on an almost-flat plane, when other edge types can't detect any line.
2. When you want full control of edge rendering. Often used for edges of squarish shapes.
3. Mark the whole base mesh to be rendered for base mesh preview.

What are edge marks not good for?

1. Round outer edges (use instead Contour/External Contour/Silhouette).

Selection by Face Marks

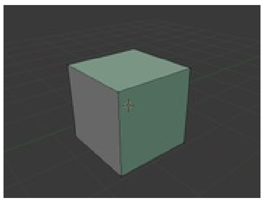

Mark Freestyle Faces.

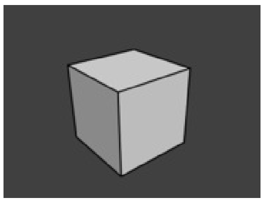
To set a face mark:

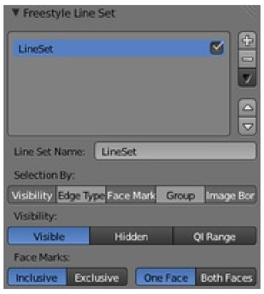
- Select a mesh and tab into Edit mode.
- Select the faces you want to be marked.
- Press CtrlF and select Mark Freestyle Face.

Face marks are useful for removing lines from certain areas of a mesh.

In this example, two faces of the default cube are marked like the image on the left. On the right is a render without face marks activated.


Marked Faces.


Render Output.



Face mark options.

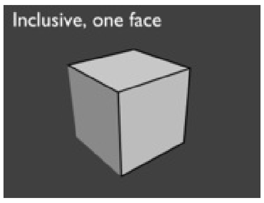
The line selection can be controlled via inclusion and faces options:

Inclusive/Exclusive
: Whether to include or exclude edges matching defined face mark conditions from the line set.

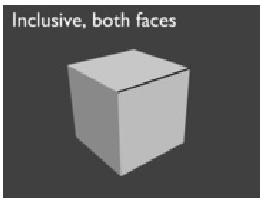
One Face
: (De)select all edges which have one or both neighbor faces marked.

Both Faces
: (De)select all edges which have both of their neighbor faces marked.

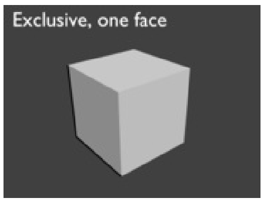
The image below shows the resulting combinations.



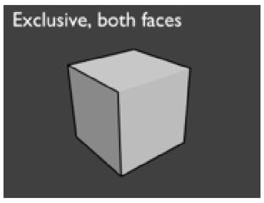
Inclusive, One Face.



Inclusive, Both Faces.



Exclusive, One Face.



Exclusive, Both Faces.

Selection by Group

You can include or exclude objects for line calculation, based on their belonging to a group.

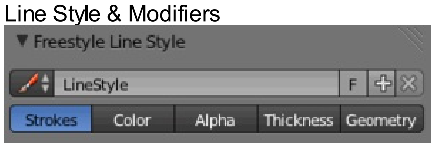
Group
: The name of the object group to use.

Inclusive/Exclusive
: Whether to include or exclude lines from those objects in this line set.

Selection by Image Border

If enabled, Freestyle only takes geometry within the image border into consideration for line calculation. This reduces render times but increases continuity problems when geometry is moved out of and into camera view.

Line Style & Modifiers

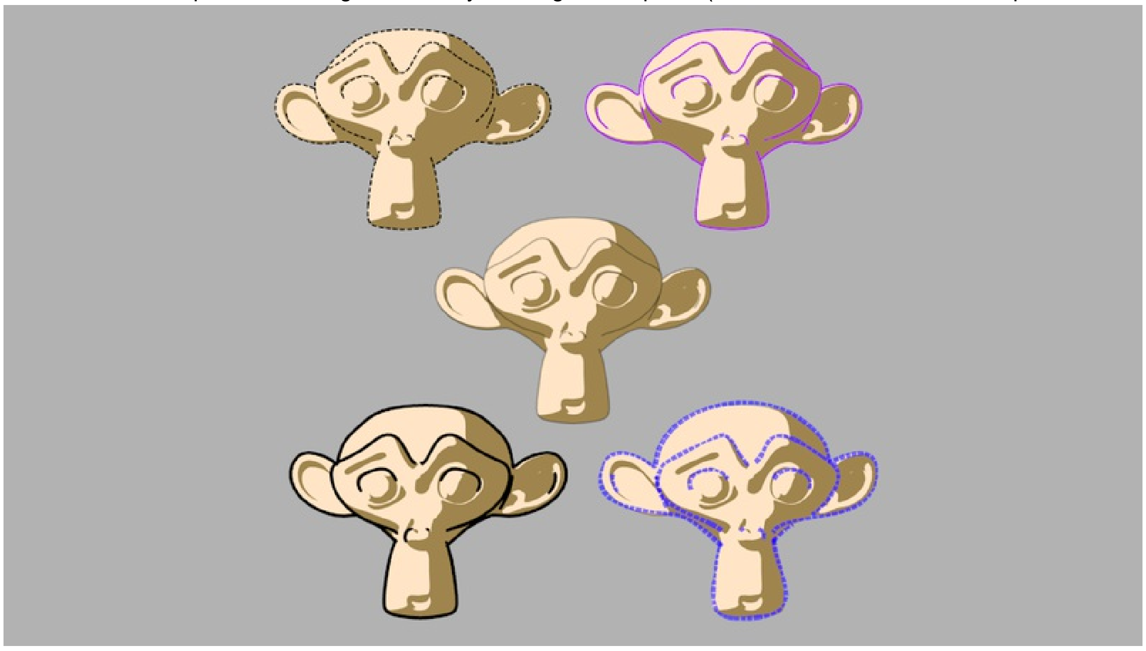


Line Style UI

In Freestyle, the line style settings define the appearance of a line set using five main aspects: stroke, color, alpha, thickness and geometry. These allow you to get many different styles of renders (technical draw, rough sketch, cartoon, oriental calligraphy, etc.).

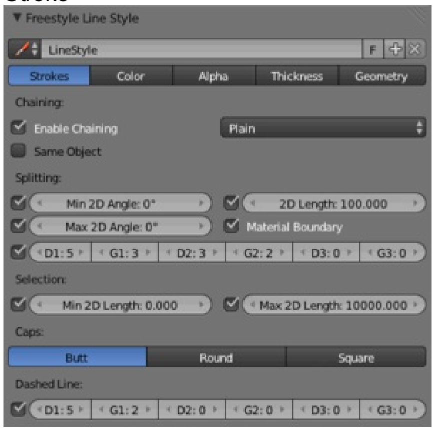
You can create as many line styles as you wish, and reuse a given line style for several line sets by selecting it from the dropdown menu next to its name.

Length Unit
Unless otherwise specified, all lengths in line style settings are in pixels (either relative or absolute, as specified in the core options).



Line Style demo File:LineStyles.zip

Stroke



Enable Chaining ✓ Plain
Same Object
Min 2D Angle: 0° ✓ 2D Length: 100.000 ✓
Max 2D Angle: 0° ✓ Material Boundary ✓
✓ D1: 5 G1: 3 D2: 3 G2: 2 D3: 0 G3: 0
Min 2D Length: 0.000 ✓ Max 2D Length: 10000.000 ✓
✓ D1: 5 G1: 2 D2: 0 G2: 0 D3: 0 G3: 0

Stroke Line Style

Strokes are the final rendered lines. Yet you can tweaks them, for example, by removing the ones longer/shorter than some threshold, chaining lines into a single stroke or breaking a stroke into several ones based on angles, dashed pattern, etc.

Chaining

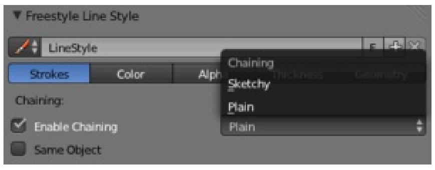
By default all retrieved lines from the line set are chained together. There are two basic chaining methods:

Plain
: The default chaining method; it creates simple chains.

Sketchy
: This chaining option allows for generating chains of feature edges with sketchy multiple strokes. Basically, it generates Round strokes instead of a single one. It is only really useful if you use some random-driven modifiers in the line style!

Rounds
: It specifies the number of rounds in sketchy strokes.

Chaining can also be turned off to render each line separately, which can be useful for line styles which depend on accurate representation of the line set.



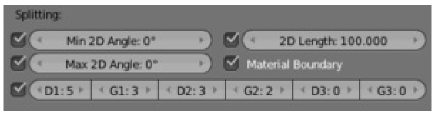
Chaining

Splitting

You can split up chains of Freestyle lines by checking one of the following:

Material Boundary
: Splits chains of feature edges if they cross from one material to another.

Min 2D Angle and Max 2D Angle
: Splits chains of feature edges when they make a 2D angle above (or below) a minimum (or maximum) threshold.



Min 2D Angle: 0° ✓ 2D Length: 100.000 ✓
Max 2D Angle: 0° ✓ Material Boundary ✓
✓ D1: 5 G1: 3 D2: 3 G2: 2 D3: 0 G3: 0

Splitting

2D Length
: Splits chains when they are longer than the given value.

D1/G1/D2/G2/D3/G3
: Splits the chains using the given dashed pattern (“D” stands for “dash”, “G” stands for “gap”; see also below).

Selection

✓ Min 2D Length: 0.000 ✓ Max 2D Length: 10000.000

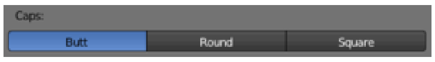
Selection

You can also choose to only select (i.e. render) chains longer than Min 2D Length and/or shorter than Max 2D Length.

Caps

You can choose between three types of line caps:

Butt
: Flat cap, exactly at the point the line ends.

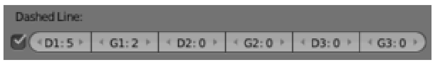


Line tip caps

Round
: A half circle centered on the end point of the line.

Square
: A square centered on the end point of the line (hence, like the circle, the drawn end of the line is slightly extended compared to its computed value).

Dashed Line



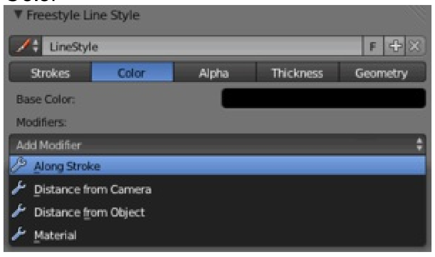
Dashes Line UI

By enabling the Dashed Line check box, you can specify three pairs of dash and gap lengths. Dash values define the lengths of dash strokes, while gap values specify intervals between two dashes.

If a zero gap is specified, then the corresponding dash is ignored even if it has a non-zero value.

Dashes are treated as separate strokes, meaning that you can apply line caps, as well as color, alpha and thickness modifiers.

Color



Line Style Color UI

In this tab you control the color of your strokes.

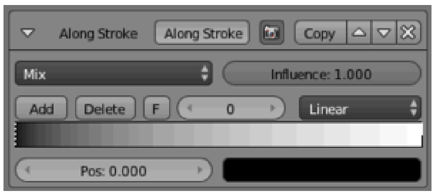
Base Color
: The base color for this line style.

Modifiers

There are four color modifiers available, which can be mixed with the base color using the usual methods (see for example the Mix compositing node for further discussion of this topic). As with other modifier stacks in Blender, they are applied from top to bottom.

Influence
: How much the result of this modifier affects the current color.

Along Stroke

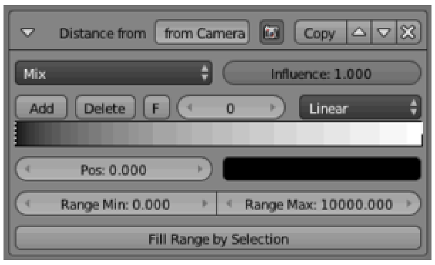


Line Style Color's Along Stroke modifier

The Along Stroke modifier alters the base color with a new one from a given color ramp mapped along each stroke's length. In other words, it applies a color ramp along each stroke.

The only settings here are those of the standard Blender color ramp!

Distance from Camera



Line Style Color's Distance From Camera modifier

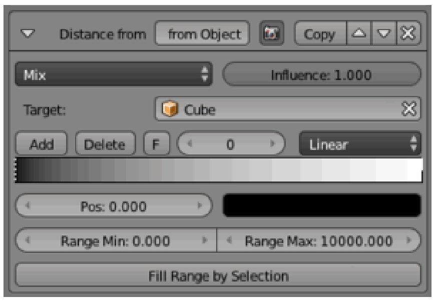
The Distance from Camera color modifier alters the base color with a new one from a given color ramp, using the distance to the active camera as the parameter.

Range Min and Range Max
: The limits of the mapping from “distance to camera” to “color in ramp”. If the current point of the stroke is at Range Min or less from the active camera, it will take the start color of the ramp, and conversely, if it is at Range Max or more from the camera, it will take the end color of the ramp. These values are in the current scene's units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected objects and the camera.

The other settings are those of the standard Blender color ramp!

Distance from Object



Line Style Color's Distance From Object modifiers

The Distance from Object color modifier alters the base color with a new one from a given color ramp, using the distance to a given object as the parameter.

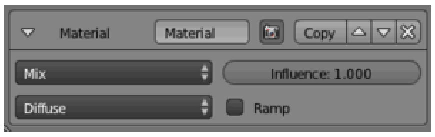
Target
: The object to measure distance from.

Range Min and Range Max
: The limits of the mapping from “distance to object” to “color in ramp”. If the current point of the stroke is at Range Min or less from the target, it will take the start color of the ramp, and conversely, if it is at Range Max or more from the target, it will take the end color of the ramp. These values are in the current scene's units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected objects and the target.

The other settings are those of the standard Blender color ramp!

Material



Line Style Color's Material modifiers

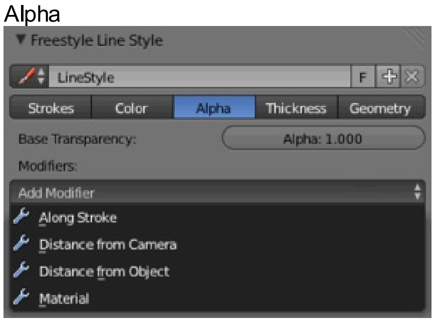
The Material color modifier alters the base color with a new one taken from the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give B&W results). In this case, an optional color ramp can be used to map these grayscale values to colored ones.

If used with the Split by Material option in the Stroke tab, the result will not be blurred between materials along the strokes.

Material modifiers demo by T.K. File:Lilies Color Material.zip

Alpha



Line Style Alpha UI

In this tab you control the alpha (transparency) of your strokes.

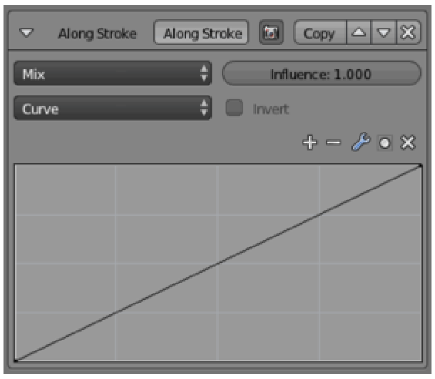
Base Transparency
: The base alpha for this line style.

Modifiers

There are four alpha modifiers available, which can be mixed with the base alpha using a subset of the usual methods (see for example the Mix compositing node for further discussion of this topic). As with other modifier stacks in Blender, they are applied from top to bottom.

Influence
: How much the result of this modifier affects the current transparency.

Along Stroke

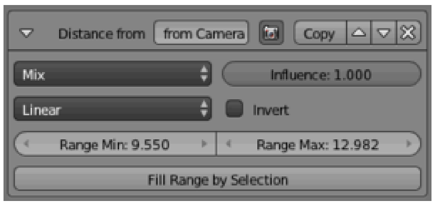


Line Style Alpha's Along Stroke modifier

The Along Stroke modifier alters the base alpha with a new one from either a linear progression or a custom curve, mapped along each stroke's length. In other words, it applies the selected progression along each stroke.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve.

Distance from Camera



Line Style Alpha's Distance From Camera modifier

The Distance from Camera modifier alters the base alpha with a new one from either a linear progression or a custom curve, using the distance to the active camera as parameter.

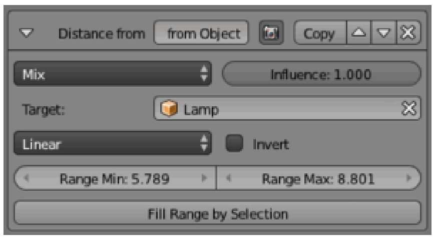
Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve.

Range Min and Range Max
: The limits of the mapping from “distance to camera” to “alpha in mapping”. If the current point of the stroke is at Range Min or

less from the active camera, it will take the start alpha of the mapping, and conversely, if it is at Range Max or more from the camera, it will take the end alpha of the mapping. These values are in the current scene's units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected objects and the camera.

Distance from Object



Line Style Alpha's Distance From Object modifier

The Distance from Object modifier alters the base alpha with a new one from either a linear progression or a custom curve, using the distance to a given object as parameter.

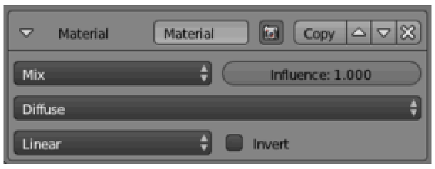
Target
: The object to measure distance from.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve.

Range Min and Range Max
: The limits of the mapping from “distance to object” to “alpha in mapping”. If the current point of the stroke is at Range Min or less from the target, it will take the start alpha of the mapping, and conversely, if it is at Range Max or more from the target, it will take the end alpha of the mapping. These values are in the current scene’s units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected objects and the target.

Material



Line Style Alpha's Material modifier

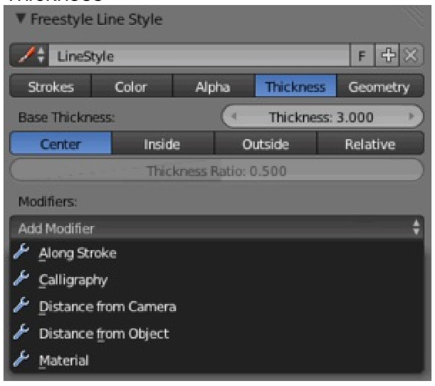
The Material modifier alters the base alpha with a new one taken from the current material under the stroke.

You can use various properties of the materials, among which some are multi-components (i.e. give RGB results). In that case, the mean value will be used.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve. Note the linear non-inverted option is equivalent to “do nothing”, as original values from materials are already in the [0.0, 1.0] range…

If used with the Split by Material option in the Stroke tab, the result will not be blurred between materials along the strokes.

Thickness



Line Style Thickness UI

In this tab you control the thickness of your strokes.

Base Thickness
: The base thickness for this line style.

Thickness Position
: Control the position of stroke thickness from the original (backbone) stroke geometry. There are four choices:

Center
: The thickness is evenly split to the left and right side of the stroke geometry.

Inside
: The strokes are drawn within object boundary.

Outside
: The strokes are drawn outside the object boundary.

Relative
: This allows you to specify the relative position by a number between **0.0** (inside) and **1.0** (outside), in the Thickness Ratio numeric field just below.

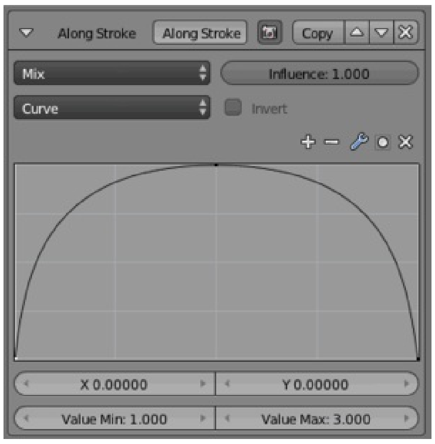
The thickness position options are applied only to strokes of edge types Silhouette and Border, since these are the only edge types defined in terms of the object boundary. Strokes of other edge types are always drawn using the Center option.

Modifiers

There are five thickness modifiers available, which can be mixed with the base thickness using a subset of the usual methods (see for example the Mix compositing node for further discussion of this topic). As with other modifier stacks in Blender, they are applied from top to bottom.

Influence
: How much the result of this modifier affects the current thickness.

Along Stroke

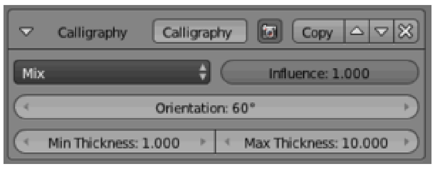


Line Style Thickness's Along Stroke modifier

The Along Stroke modifier alters the base thickness with a new one from either a linear progression or a custom curve, mapped along each stroke's length. In other words, it applies the selected progression along each stroke.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve.

Calligraphy

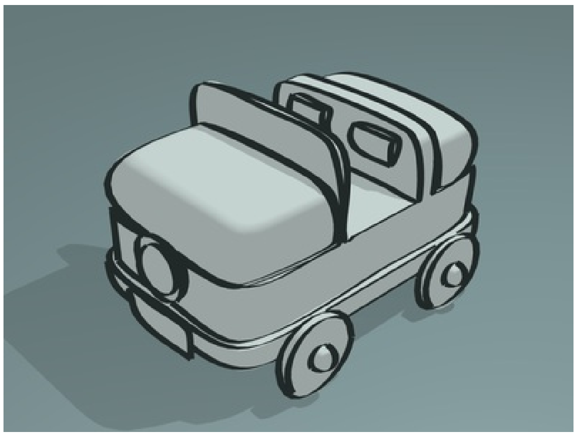


Line Style Thickness's Calligraphy modifier

The Calligraphy modifier mimics some broad and flat pens for calligraphy. It generates different thickness based on the orientation of the stroke.

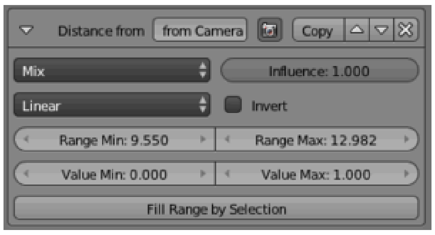
Orientation
: The angle (orientation) of the virtual drawing tool, from the vertical axis of the picture. For example, an angle of **0.0°** mimics a pen aligned with the vertical axis, hence the thickest strokes will be the vertical ones, and the thinnest, the horizontal ones.

Min Thickness and Max Thickness
: The minimum and maximum generated thickness (as explained above, minimum is used when the stroke’s direction is perpendicular to the main Orientation, and maximum, when aligned with it).



Calligraphy modifier demo by T.K. File:Toycar Calligraphy.zip

Distance from Camera



Line Style Thickness's Distance From Camera modifier

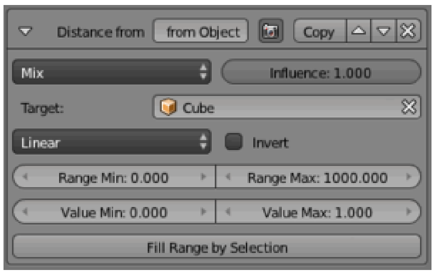
The Distance from Camera modifier alters the base thickness with a new one from either a linear progression or a custom curve, using the distance to the active camera as the parameter.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve.

Range Min and Range Max
: The limits of the mapping from “distance to camera” to “thickness in mapping”. If the current point of the stroke is at Range Min or less from the active camera, it will take the start thickness of the mapping, and conversely, if it is at Range Max or more from the camera, it will take the end thickness of the mapping. These values are in the current scene’s units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected objects and the camera.

Distance from Object



Line Style Thickness's Distance from Object modifier

The Distance from Object modifier alters the base thickness with a new one from either a linear progression or a custom curve, using the distance to a given object as parameter.

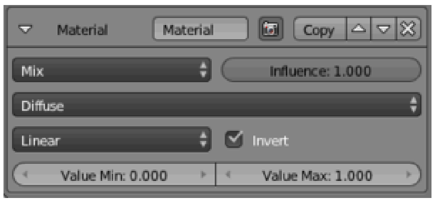
Target
: The object to measure distance from.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve.

Range Min and Range Max
: The limits of the mapping from “distance to object” to “alpha in mapping”. If the current point of the stroke is at Range Min or less from the target, it will take the start thickness of the mapping, and conversely, if it is at Range Max or more from the target, it will take the end thickness of the mapping. These values are in the current scene’s units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected objects and the target.

Material



Line Style Thickness's Material modifier

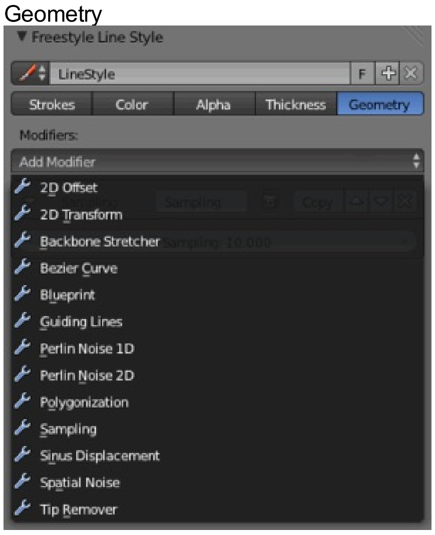
The Material modifier alters the base thickness with a new one taken from the current material under the stroke.

You can use various properties of the materials, among which some are multi-components (i.e. give RGB results). In that case, the mean value will be used.

Mapping
: Either a linear progression (from **0.0** to **1.0**, which may be inverted with the Invert option), or a custom mapping curve. Note the linear non-inverted option is equivalent to “do nothing”, as original values from materials are already in the [0.0, 1.0] range…

If used with the Split by Material option in the Stroke tab, the result will not be blurred between materials along the strokes.

Geometry



Line Style Geometry Overall UI

In this tab you control the geometry of your strokes.

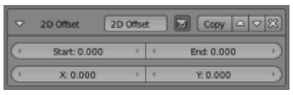
Modifiers

There are thirteen geometry modifiers available. These modifiers have no mix nor influence settings, as they always completely apply to the strokes' geometry (like object modifiers do). They take the resulting two-dimensional strokes from the Freestyle line set and displace or deform them in various ways.

As with other modifier stacks in Blender, they are applied from top to bottom.

2D Offset

The 2D Offset modifier adds some two-dimensional offsets to the stroke backbone geometry. It has two sets of independent options/effects:



2D Offset 2D Offset Copy

Start: 0.000 End: 0.000

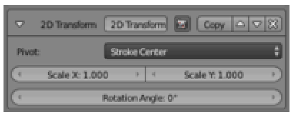
X: 0.000 Y: 0.000

Line Style Geometry's 2D Offset modifier

Start and End
: These two options add the given amount of offset to the start (or end) point of the stroke, along the (2D) normal at those points. The effect is blended over the whole stroke, so if you, for example, set only Start to **50**, the start of the stroke is offset 50 pixels along its normal, the middle of the stroke, 25 pixels along its own normal, and the end point isn't moved.

X and Y
: These two options simply add a constant horizontal and/or vertical offset to the whole stroke.

2D Transform



2D Transform 2D Transform Copy

Pivot: Stroke Center

Scale X: 1.000 Scale Y: 1.000

Rotation Angle: 0°

Line Style Geometry's 2D Transform modifier

The 2D Transform modifier applies two-dimensional scaling and/or rotation to the stroke backbone geometry. Scale is applied before rotation.

The center (pivot point) of these 2D transformations can be:

Stroke Center
: The median point of the stroke.

Stroke Start
: The beginning point of the stroke.

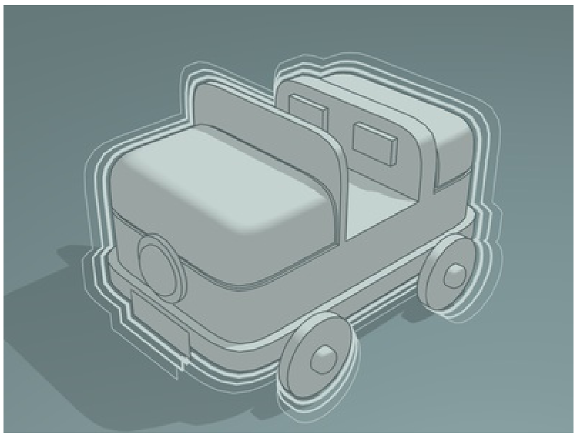
Stroke End
: The end point of the stroke.

Stroke Point Parameter
: The Stroke Point Parameter factor controls where along the stroke the pivot point is (**0.0** means start point; **1.0** end point).

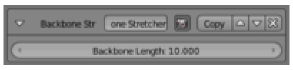
Absolute 2D Point
: The Pivot X and Pivot Y allows you to define the position of the pivot point in the final render (from the bottom left corner). **WARNING**: Currently, you have to take into account the *real* render size, i.e. resolution **and** resolution percentage!

Scale X and Scale Y
: The scaling factors, in their respective axes.

Rotation Angle
: The rotation angle.


2D Transform modifier File:Toycar Three Contours.zip

Backbone Stretcher



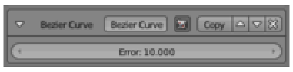
Backbone Length: 10.000

Line Style Geometry's
Backbone Stretcher modifier

The Backbone Stretcher modifier stretches (adds some length to) the beginning and end of the stroke.

Backbone Length
: Length to add to the strokes’ ends.

Bezier Curve

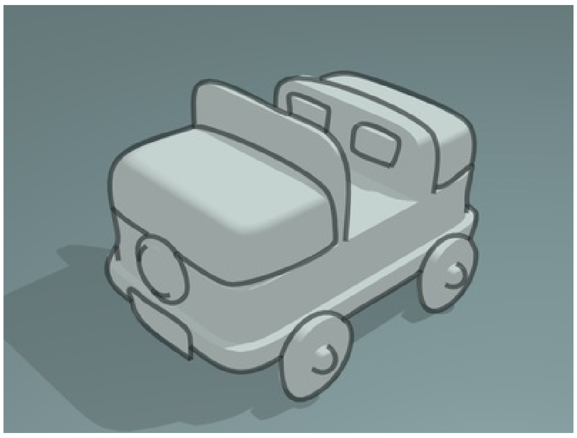


Error: 10.000

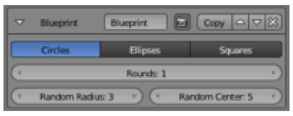
Line Style Geometry's Bezier
Curve modifier

The Bezier Curve modifier replaces the stroke by a Bezier approximation of it.

Error
: The maximum distance allowed between the new Bezier curve and the original stroke.


Bezier Curve modifier demo by T.K. File:Toycar bezier.zip

Blueprint


Blueprint
Circles Ellipses Squares
Rounds: 1
Random Radius: 3
Random Center: 5

Line Style Geometry's
Blueprint modifier

The Blueprint modifier produces blueprint-like strokes using either circular, elliptical, or square contours. A blueprint here refers to those lines drawn at the beginning of free-hand drawing to capture the silhouette of objects with a simple shape such as circles, ellipses and squares.

Shape
: Which base shapes to use for this blueprint: Circles, Ellipses or Squares.

Rounds
: How many rounds are generated, as if the pen draws the same stroke several times (i.e. how many times the process is repeated).

Random Radius and Random Center
: For the Circles and Ellipses shapes. Adds some randomness to each round in the relevant aspect. Using more than one round with no randomness would be meaningless, as they would draw over each other exactly.

Backbone Length and Random Backbone
: For the Squares shapes. The first adds some extra length to each edge of the generated squares (also affected by the second parameter). The second adds some randomness to the squares.

Note that the Min 2D Length feature from the Strokes settings is quite handy here, to avoid the noise generated by small strokes…

Guiding Lines

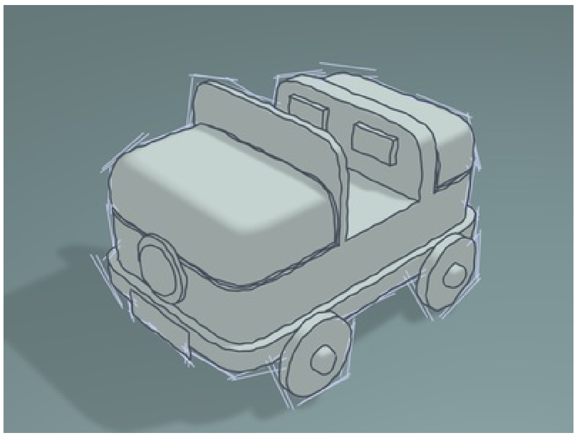
Guiding Lines
Offset: 0.000

Line Style Geometry's
Guiding Lines modifier

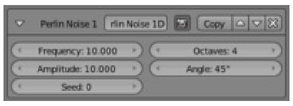
The Guiding Lines modifier replaces a stroke by a straight line connecting both of its ends.

Offset
: Offset the start and end points along the original stroke, before generating the new straight one.

This modifier will produce reasonable results when strokes are short enough, because shorter strokes are more likely to be well approximated by straight lines. Therefore, it is recommended to use this modifier together with one of the splitting options (by 2D angle or by 2D length) from the Strokes panel.


Guiding Lines modifier Demo by T.K. File:Toycar Guiding Line.zip

Perlin Noise 1D


Perlin Noise 1
Frequency: 10.000
Octaves: 4
Amplitude: 10.000
Angle: 45°
Seed: 0

Line Style Geometry's Perlin
Noise 1D modifier

The Perlin Noise 1D modifier adds one-dimensional Perlin noise to the stroke.

Frequency
: How dense the noise is (kind of a scale factor along the stroke).

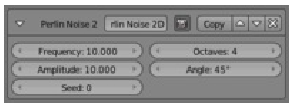
Amplitude
: How much the noise distorts the stroke in the Angle direction.

Seed
: The seed of the random generator (the same seed over a stroke will always give the same result).

Octaves
: The “level of detail” of the noise.

Angle
: In which direction the noise is applied (**0.0°** is fully horizontal).

Perlin Noise 2D



Frequency: 10.000
Octaves: 4
Amplitude: 10.000
Angle: 45°
Seed: 0

Line Style Geometry's Perlin Noise 2D modifier

The Perlin Noise 2D modifier adds one-dimensional Perlin noise to the stroke.

Its settings are exactly the same as the Perlin Noise 1D modifier.

TODO: What’s the difference between those two modifiers?

Polygonization

Error: 10.000

Line Style Geometry's Polygonization modifier

The Poligonization modifier simplifies strokes as much as possible (in other words, it transforms smooth strokes into jagged polylines).

Error
: The maximum distance allowed between the new simplified stroke and the original one (the larger this value is, the more jagged/approximated the resulting polylines are).

Sampling

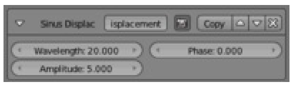
Sampling: 10.000

Line Style Geometry's Sampling modifier

The Sampling modifier changes the definition, precision of the stroke, for the following modifiers.

Sampling
: The smaller this value, the more precise are the strokes. Be careful; too small values will require a huge amount of time and memory during render!

Sinus Displacement



Wavelength: 20.000
Phase: 0.000
Amplitude: 5.000

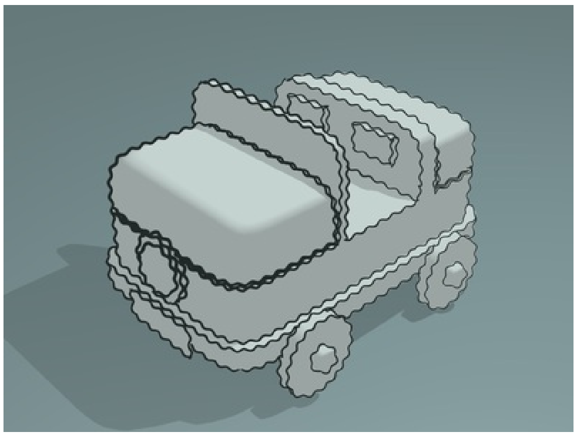
Line Style Geometry's Sinus Displacement modifier

The Sinus Displacement modifier adds a sinusoidal displacement to the stroke.

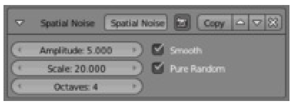
Wavelength
: How wide the undulations are along the stroke.

Amplitude
: How high the undulations are across the stroke.

Phase
: Allows “offsetting” (“moving”) the undulations along the stroke.


Sinus Displacement modifier demo by T.K. File:Toycar Sinus.zip

Spatial Noise



Line Style Geometry's Spatial Noise modifier

The Spatial Noise modifier adds some spatial noise to the stroke.

TODO: definition of "spatial noise"!

Amplitude
: How much the noise distorts the stroke.

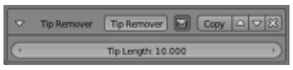
Scale
: How wide the noise is along the stroke.

Octaves
: The level of detail of the noise.

Smooth
: When enabled, apply some smoothing over the generated noise.

Pure Random
: When disabled, the next generated random value depends on the previous one; otherwise they are completely independent. Disabling this setting gives a more “consistent” noise along a stroke.

Tip Remover



Line Style Geometry's Tip Remover modifier

The Tip Remover modifier removes a piece of the stroke at its beginning and end.

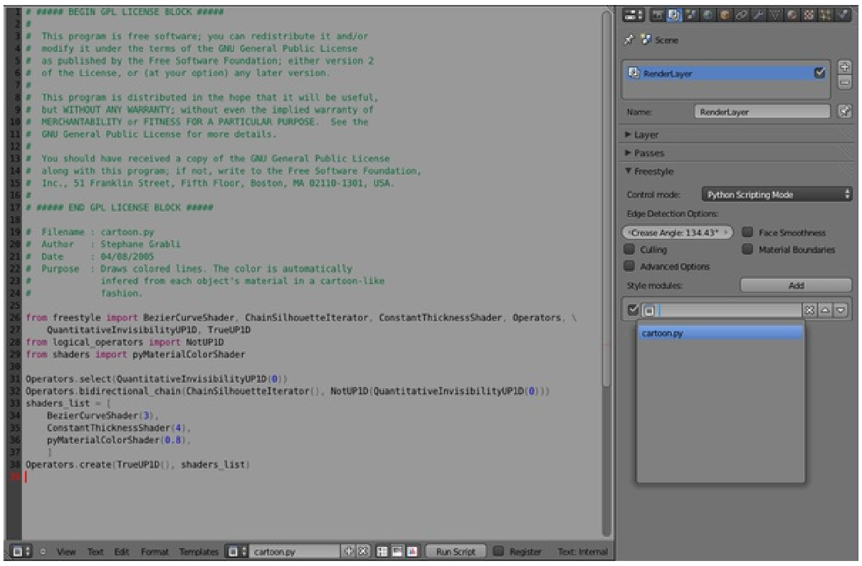
Tip Length
: Length of stroke to remove at both of its tips.

Python Scripting Mode

The Python Scripting mode offers full programmability for line stylization. In this control mode, all stylization operations are written as Python scripts referred to as style modules in the Freestyle terminology. The input to a style module is a view map (i.e., a set of detected feature edges), and the output is a set of stylized strokes.

A style module is composed of successive calls of five basic operators: selection, chaining, splitting, sorting and stroke creation. The selection operator identifies a subset of input feature edges based on one or more user-defined selection conditions (predicates). The selected edges are processed with the chaining, splitting and sorting operators to build chains of feature edges. These operators are also controlled by user-supplied predicates and functions in order to determine how to transform the feature edges into chains. Finally, the chains are transformed into stylized strokes by the stroke creation operator, which takes a list of user-defined stroke shaders.

Python style modules are stored within .blend files as text datablocks. External style module files first need to be loaded in the Text Editor window. Then the pull-down menu within an entry of the style module stack allows you to select a module from the list of loaded style modules.



```
# ##### BEGIN GPL LICENSE BLOCK #####
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software Foundation,
#  Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
#
# ##### END GPL LICENSE BLOCK #####

#  Filename : cartoon.py
#  Author   : Stephane Grabli
#  Date     : 04/08/2005
#  Purpose  : Draws colored lines. The color is automatically
#             infered from each object's material in a cartoon-like
#             fashion.

from freestyle import BezierCurveShader, ChainSilhouetteIterator, ConstantThicknessShader, Operators, \
    QuantitativeInvisibilityUP1D, TrueUP1D
from logical_operators import NotUP1D
from shaders import pyMaterialColorShader

Operators.select(QuantitativeInvisibilityUP1D(0))
Operators.bidirectional_chain(ChainSilhouetteIterator(), NotUP1D(QuantitativeInvisibilityUP1D(0)))
shaders_list = [
    BezierCurveShader(3),
    ConstantThicknessShader(4),
    pyMaterialColorShader(0.8),
    ]
Operators.create(TrueUP1D(), shaders_list)
```

Name: RenderLayer
Control mode: Python Scripting Mode
Crease Angle: 134.43°
Style modules: cartoon.py

A screen capture of a style module (cartoon.py) loaded in the Text Editor window (left), as well as Freestyle options in the Python Scripting mode in the Render Layers buttons (right)

Freestyle for Blender comes with a number of Python style modules that can serve as a starting point of your own style module writing. See also the section of the Freestyle Python API in the Blender Python API reference manual for the full detail of style module constructs.

By T.K. using the Python Scripting mode (File:Turning Pages.zip, CC0)

By T.K. using the Python Scripting mode (File:Lily Broken Topology.zip, CC0)

Writing Style Modules

A style module is a piece of code responsible for the stylization of Freestyle line drawing. The input of a style module is a set of feature edges called view map (ViewMap). The output is a set of stylized lines also referred to as strokes. A style module is structured as a pipeline of operations that allow for building strokes from the input edges within the view map. There are five kinds of operations (corresponding operator functions in parentheses):

- Selection (Operators.select())
- Chaining (Operators.chain(), Operators.bidirectional_chain())
- Splitting (Operators.sequential_split(), Operators.recursive_split())
- Sorting (Operators.sort())
- Stroke creation (Operators.create())

The input view map is populated with a set of ViewEdge objects. The selection operation is used to pick up ViewEdges of interest to artists based on user-defined selection conditions (predicates). Chaining operations take the subset of ViewEdges and build Chains by concatenating ViewEdges according to user-defined predicates and functions. The Chains can be further refined by splitting them into smaller pieces (e.g., at points where edges make an acute turn) and selecting a fraction of them (e.g., to keep only those longer than a length threshold). The sorting operation is used to arrange the stacking order of chains to draw one line on top of another. The chains are finally transformed into stylized strokes by the stroke creation operation applying a series of stroke shaders to individual chains.

ViewEdges, Chains and Strokes are generically referred to as one-dimensional (1D) elements. A 1D element is a polyline that is a series of connected straight lines. Vertices of 1D elements are called 0D elements in general.

All the operators act on a set of active 1D elements. The initial active set is the set of ViewEdges in the input view map. The active set is updated by the operators.

Selection

The selection operator goes through every element of the active set and keeps only the ones satisfying a certain predicate. The Operators.select() method takes as the argument a unary predicate that works on any Interface1D that represents a 1D element. For example:

```
Operators.select(QuantitativeInvisibilityUP1D(0))
```

This selection operation uses the QuantitativeInvisibilityUP1D predicate to select only the visible ViewEdge (more precisely, those whose quantitative invisibility is equal to 0). The selection operator is intended to selectively apply the style to a fraction of the active 1D elements.

It is noted that QuantitativeInvisibilityUP1D is a class implementing the predicate that tests line visibility, and the Operators.select() method takes an instance of the predicate class as argument. The testing of the predicate for a given 1D element is actually done by calling the predicate instance, that is, by invoking the __call__ method of the predicate class. In other words, the Operators.select() method takes as argument a functor which in turn takes an Interface0D object as argument. The Freestyle Python API employs functors extensively to implement predicates, as well as functions.

Chaining

The chaining operators act on the set of active ViewEdge objects and determine the topology of the future strokes. The idea is to implement an iterator to traverse the ViewMap graph by marching along ViewEdges. The iterator defines a chaining rule that determines the next ViewEdge to follow at a given vertex (see ViewEdgeIterator). Several such iterators are provided as part of the Freestyle Python API (see ChainPredicateIterator and ChainSilhouetteIterator). Custom iterators can be defined by inheriting the ViewEdgeIterator class. The chaining operator also takes as argument a UnaryPredicate working on Interface1D as a stopping criterion. The chaining stops when the iterator has reached a ViewEdge satisfying this predicate during the march along the graph.

Chaining can be either unidirectional (Operators::chain()) or bidirectional (Operators::bidirectional_chain()). In the latter case, the chaining will propagate in the two directions from the starting edge.

The following is a code example of bidirectional chaining:

```
Operators.bidirectional_chain(ChainSilhouetteIterator(),
                              NotUP1D(QuantitativeInvisibilityUP1D(0)))
```

The chaining operator uses the ChainSilhouetteIterator as the chaining rule and stops chaining as soon as the iterator has come to an invisible ViewEdge.

The chaining operators process the set of active ViewEdge objects in order. The active ViewEdges can be previously sorted using the Operators::sort() method (see below). It starts a chain with the first ViewEdge of the active set. All ViewEdges that have already been involved in the chaining process are marked (in the case of the example above, the time stamp of each ViewEdge is modified by default), in order not to process the same ViewEdge twice. Once the chaining reaches a ViewEdge that satisfies the stopping predicate, the chain is terminated. Then a new chain is started from the first unmarked ViewEdge in the active set. This operation is repeated until the last unmarked ViewEdge of the active set was processed. At the end of the chaining operation, the active set is set to the Chains that have just been constructed.

Splitting

The splitting operation is used to refine the topology of each Chain. Splitting is performed either sequentially or recursively. Sequential splitting (Operators::sequentialSplit()) in its basic form, parses the Chain at a given arbitrary resolution and evaluates a unary predicate (working on 0D elements) at each point along the Chain. Every time the predicate is satisfied, the chain is split into two chains. At the end of the sequential split operation, the active set of chains is set to the new chains.

```
Operators.sequentialSplit(TrueUP0D(), 2)
```

In this example, the chain is split every 2 units. A more elaborated version uses two predicates instead of one: One to determine the starting point of the new chain and the other to determine its ending point. This second version can lead to a set of Chains that are disjoint or that overlap if the two predicates are different. (see Operators::sequentialSplit() for more details).

Recursive splitting (Operators::recursiveSplit()) evaluates a function on the 0D elements along the Chain at a given resolution and find the point that gives the maximum value for the function. The Chain is then split into two at that point. This process is recursively repeated on each of the two new Chains, until the input Chain satisfies a user-specified stopping condition.

```
func = Curvature2DAngleF0D()
Operators.recursive_split(func, NotUP1D(HigherLengthUP1D(5)), 5)
```

In the code example above, the Chains are recursively split at points of the highest 2D curvature. The curvature is evaluated at points along the Chain at a resolution of 5 units. Chains shorter than 5 units won't be split anymore.

Sorting

The sorting operator (Operators::sort()) arranges the stacking order of active 1D elements. It takes as argument a binary predicate used as a “smaller than” operator to order two 1D elements.

```
Operators.sort(Length2DBP1D())
```

In this code example, the sorting uses the Length2DBP1D binary predicate to sort the Interface1D objects in the ascending order in terms of 2D length.

The sorting is particularly useful when combined with causal density. Indeed, the causal density evaluates the density of the resulting image as it is modified. If we wish to use such a tool to decide to remove strokes whenever the local density is too high, it is important to control the order in which the strokes are drawn. In this case, we would use the sorting operator to insure that the most "important" lines are drawn first.

Stroke creation

Finally, the stroke creation operator (Operators::create()) takes the active set of Chains as input and build Strokes. The operator takes two arguments. The first is a unary predicate that works on Interface1D that is designed to make a last selection on the set of chains. A Chain that doesn't satisfy the condition won't lead to a Stroke. The second input is a list of Shaders that will be responsible for the shading of each built stroke.

```
shaders_list = [
    SamplingShader(5.0),
    ConstantThicknessShader(2),
    ConstantColorShader(0.2,0.2,0.2,1),
    ]
Operators.create(DensityUP1D(8,0.1, IntegrationType.MEAN), shaders_list)
```

In this example, the DensityUP1D predicate is used to remove all Chains whose mean density is higher than 0.1. Each chain is transformed into a stroke by resampling it so as to have a point every 5 units and assigning to it a constant thickness of 2 units and a dark gray constant color.

User control on the pipeline definition

Style module writing offers different types of user control, even though individual style modules have a fixed pipeline structure. One is the sequencing of different pipeline control structures, and another is through the definition of functor objects that are passed as argument all along the pipeline.

Different pipeline control structures can be defined by sequencing the selection, chaining, splitting, and sorting operations. The stroke creation is always the last operation that concludes a style module.

Predicates, functions, chaining iterators, and stroke shaders can be defined by inheriting base classes and overriding appropriate methods. See the reference manual entries of the following base classes for more information on the user-scriptable constructs.

- UnaryPredicate0D
- UnaryPredicate1D
- BinaryPredicate0D
- BinaryPredicate1D
- UnaryFunction0DDouble
- UnaryFunction0DEdgeNature
- UnaryFunction0DFloat
- UnaryFunction0DId
- UnaryFunction0DMaterial
- UnaryFunction0DUnsigned
- UnaryFunction0DVec2f
- UnaryFunction0DVec3f
- UnaryFunction0DVectorViewShape
- UnaryFunction0DViewShape
- UnaryFunction1DDouble
- UnaryFunction1DEdgeNature
- UnaryFunction1DFloat
- UnaryFunction1DUnsigned
- UnaryFunction1DVec2f
- UnaryFunction1DVec3f
- UnaryFunction1DVectorViewShape
- UnaryFunction1DVoid
- ViewEdgeIterator
- StrokeShader

Links

Here are some links to external data regarding Freestyle.

Videos

[video link]
The Light At The End

[video link]
mmd_tools test2 with Blender+Freestyle (未来時計 AM4:30)

Video Tutorials

[video link]
An introduction to Freestyle plugin for Blender : "sketching" Suzanne / HD

[video link]
Using freestyle in blender

[video link]
Tutorial: Blender 3D - Freestyle and Composite

[video link]
Blender Tutorial: Freestyle

Tutorials

Freestyle basics
http://studiollb.wordpress.com/2012/02/29/freestyle-introductory-tutorial/
http://jikz.net/archives/364
http://jikz.net/archives/329

Edge types
https://studiollb.wordpress.com/2012/09/08/freestyle-101-edge-types/

Line style basic
http://studiollb.wordpress.com/2012/09/08/freestyle-101-line-style-basic/

Line style modifiers
http://studiollb.wordpress.com/2012/09/08/freestyle-101-line-style-modifier-part-1/
http://studiollb.wordpress.com/2012/09/08/freestyle-101-line-style-modifier-part-2/
http://studiollb.wordpress.com/2012/09/15/freestyle-101-planning-and-along-stroke-line-style-modifier/

Tips and tricks
http://studiollb.wordpress.com/2012/02/03/freestyle-tips/ (Old)

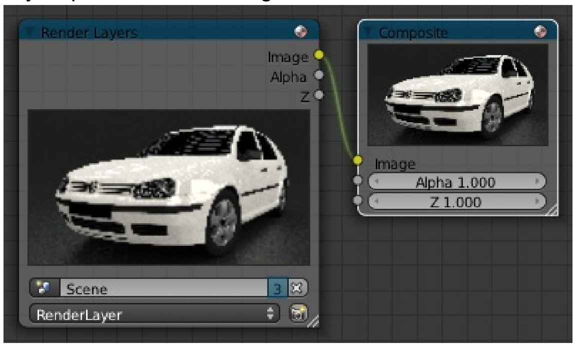
Misc

- FreeStyle Users’ improvement suggestions.
- FreeStyle integration into Blender blog
- Early documentation of FreeStyle

Composite Nodes

Compositing Nodes allow you to assemble and enhance an image (or movie). Using composition nodes, you can glue two pieces of footage together and colorize the whole sequence all at once. You can enhance the colors of a single image or an entire movie clip in a static manner or in a dynamic way that changes over time (as the clip progresses). In this way, you use composition nodes to both assemble video clips together, and enhance them.

Term: Image
We use the term *Image* to refer to a single picture, a picture in a numbered sequence of images, or a frame of a movie clip. A node layout processes one image at a time, no matter what kind of input you provide.



To process your image, you use nodes to import the image into Blender, change it, optionally merge it with other images, and finally save it.

The example to the right shows the simplest noodle; an input node threads the camera view to an output node so it can be saved.

Nodes Concepts

Nodes

"Nodes" are individual blocks that perform a certain operation, and might have one or many different outputs.

Conceptually, there are three basic types of nodes:

- **Input Nodes**

  these nodes *produce* information, but do not have any inputs of their own.
  Examples are: *Render Layers*, *Value* and *RGB* nodes.

- **Processing Nodes**:

  these nodes *filter* or *transform* their inputs, to produce one or more outputs.
  Examples are: *RGB Curves*, *Defocus,'* and **Vector Blur** nodes.

- **Output Nodes**:

  these nodes *consume* their inputs to produce some kind of meaningful result.
  Examples are: *Composite* node (which determines the final output used by Blender), *Viewer* (which displays the output of a socket), and **File Output** node.

Noodles

The essential idea of nodes is that you can create an arbitrarily-complex *network* of nodes, by connecting the *outputs* of one or more nodes to the *inputs* of one or more other nodes. Then, you can set appropriate parameters (as you see fit) for each node.

This network is called a "noodle" and it describes how information literally *flows through* to produce whatever result you want.

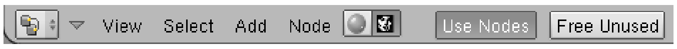
Node Groups

You can define *node groups*, and use those groups as they were a single node.

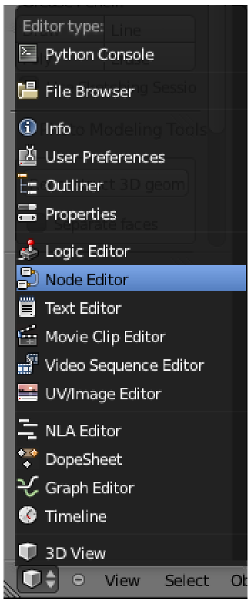
You can link and append these node groups from other files.

Accessing and Activating Nodes

Access the Node Editor and enable Composite Nodes by clicking on the *Image* icon.

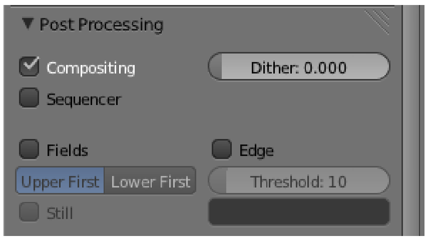


Node Editor Header with Composite Nodes enabled



Select the Node Editor window

To activate nodes for compositing, click the Use Nodes checkbox. Blender creates a default starting noodle, consisting of two nodes threaded together.



Use Composition Nodes

To use this mini-map, you must now tell Blender to use the Compositing Node map that has been created, and to composite the image using composition nodes. To do so, switch to the Render button area and activate the **Compositing** button located below the Post Processing tab. This tells Blender to composite the final image by running it through the composition node map.

You now have your first noodle, a RenderLayer input node threaded to a Composite output node. From here, you can add and connect many types of compositing nodes, in a sort of map layout, to your heart's content (or physical memory constraints, whichever comes first).

Examples

You can do just about anything with images using nodes.

Raw footage from a foreground actor in front of a blue screen, or a rendered object doing something, can be layered on top of a background. Composite both together, and you have composited footage.

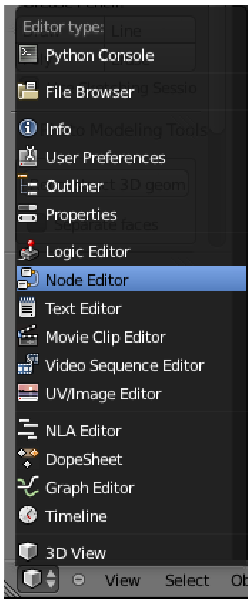
You can change the mood of an image:

- To make an image 'feel' colder, a blue tinge is added.
- To convey a flashback or memory, the image may be softened.
- To convey hatred and frustration, add a red tinge or enhance the red. The film 'Sin City' is the most extreme example of this I have ever seen.
- A startling event may be sharpened and contrast-enhanced.
- A happy feeling - you guessed it - add yellow (equal parts red and green, no blue) for bright and sunny.
- Dust and airborne dirt is often added as a cloud texture over the image to give a little more realism.

The Node Editor

This section explains the window in general, and its header menu options. It also tells you how to enable nodes for use within Blender.

Accessing The Node Editor

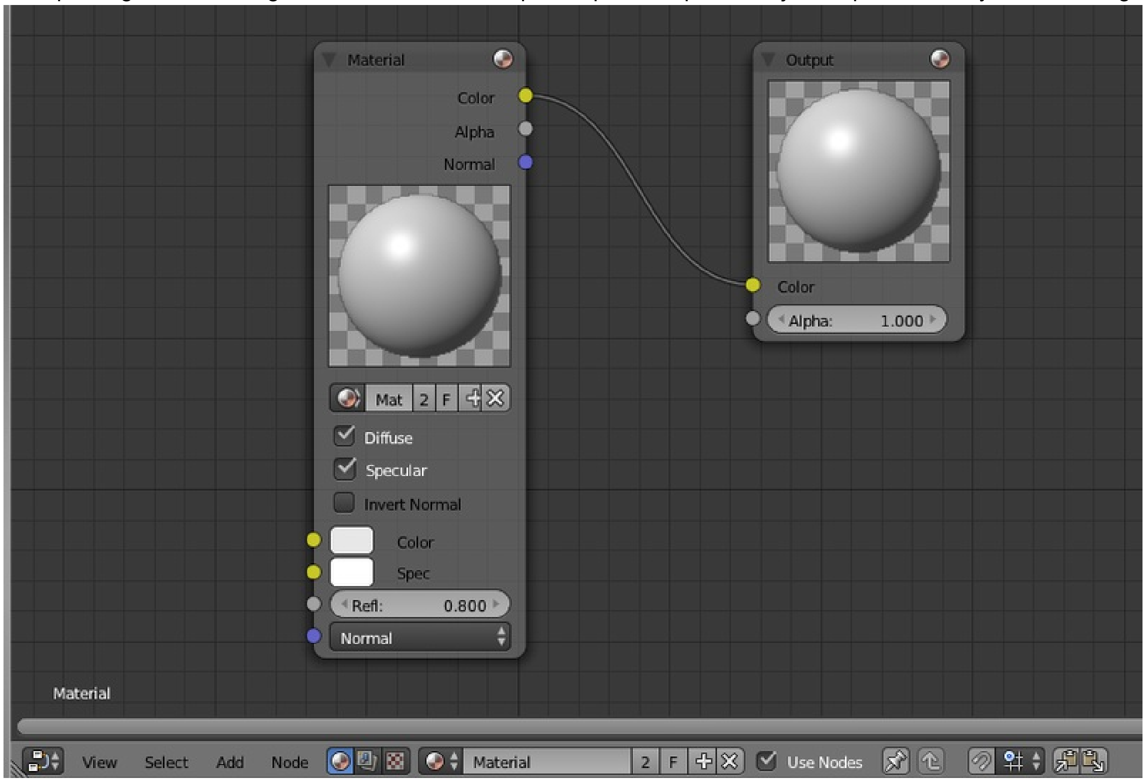


Select the Node Editor window.

First let's enter the node editor by changing our window type to Node Editor. As shown in *Select the Node Editor window*, click on the window type icon and select Node Editor from the popup list. Node maps can get quite large, so use or create a big window. The window has a graph-paper style background and a header.

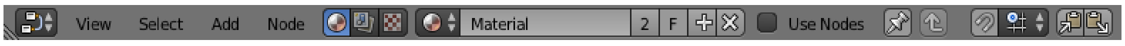
Each scene within your blend file can have multiple Material Node maps and ONE Compositing Node map. The Node Editor window shows either type of map, depending on the selector position.

Hint
You might want to add a new window layout called 6-Nodes (the list is shown on the User Preferences header at the top of your screen) comprised mostly of one big Node Editor window. My layout has the buttons window at the bottom and a text editor window on the side for me to keep notes. If you have a widescreen display (or even a regular one), you might also want to add a 3D view or UV/Image Editor window to the left side of the Node window layout, so you can work with images or your model while you're manipulating nodes. Having the 3D Preview Render panel open on top of an object is quite useful if you're tweaking material nodes.



Node Editor.

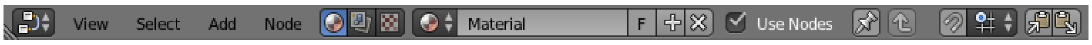
By default, the header, when first displayed, is uninitialized as shown:



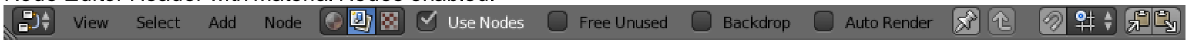
Default Node Editor header.

Activating Nodes

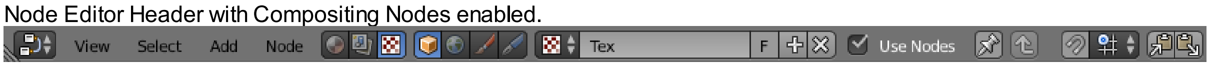
- What nodes to use?
  - If you want to work with a material node map, click the ball in the Material/Compositing node set selector. (See *Node Editor Header with Material Nodes enabled.*)
  - If you want to work with a compositing node map, click the overlaped pictures on the Material/Compositing node set selector. (See *Node Editor Header with Compositing Nodes enabled.*)
  - If you want to work with a texture node map, click the checker on the Material/Compositing node set selector. (See *Node Editor Header with Texture Nodes enabled.*)
- To actually activate nodes, click the Use Nodes button.
- The first time that you select either a Material, Compositing or a Texture node map, the Node Editor window will be instantly filled with starter input and output compositing nodes already connected together.



Node Editor Header with Material Nodes enabled.



Node Editor Header with Compositing Nodes enabled.



Node Editor Header with Texture Nodes enabled.

Node Editor Window Actions

When the cursor is in the window, several standard Blender hotkeys and mouse actions are available, including:

Popup menu
: Space - Brings up a main popup menu, allowing you to add, view, select, etc.

Delete
: X or Del - Deletes the selected node(s).

Box select
: B - Starts the bounding box selection process. Position your cursor and LMB click & drag to select a set of nodes.

Cut connections (lasso)
: CtrlAlt LMB click & drag - Starts a lasso selection, BUT when you let up the mouse button, all threads (connections) within the lasso are broken.

Undo
: CtrlZ Very helpful if you forgot to press B before box-selecting, eh?

Redo
: CtrlY or Ctrl⇧ ShiftZ - You can use this if you used "undo" a bit too often :)

Select multiple
: ⇧ Shift LMB or ⇧ Shift RMB - Multiple node select.

Grab/Move
: G - Moves your current selection around.

Standard Window Control
Node maps can get pretty hairy (large and complicated, that is). The contents of the window (the node map) can be panned just like any other Blender window by clicking MMB and dragging about. Wheeling Wheel up/down or using the keypad + NumPad/- NumPad will zoom in/out. The window can be resized and combined using the standard window techniques (see *Navigating in 3d Space*).

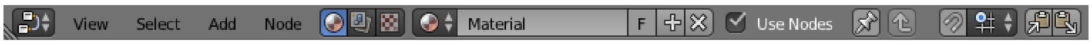
Node Editor Header

At a glance

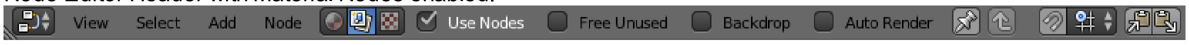
On the window header, you will see header options:

- View - to see things more clearly;
- Select - to do things more clearly;
- Add - to walk with...err..to add Nodes, organized by type;
- Node - to do things with selected nodes, akin to vertices;
- a Material, Compositing or Texture node set selector;
- a Use Nodes button;
- a Use Pinned button;
- a Go to Parent button;
- a Snap button;
- a Snap Node Element selector;

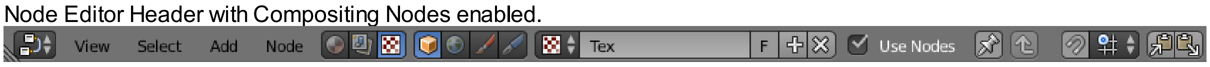
- a Copy Nodes button;
- a Paste Nodes button.



Node Editor Header with Material Nodes enabled.



Node Editor Header with Compositing Nodes enabled.



Node Editor Header with Texture Nodes enabled.

Menus

View, Select and Add

These popup menus provide the basic functions:

View
: This menu changes your view of the window, standing in for the standard keyboard shortcuts + NumPad (zoom in), - NumPad (zoom out), ↖ Home (zoom all) or equivalent mouse actions.

Select
: This menu allows you to select a node or groups of nodes, and does the same as typing the hotkey to select all A or start the border select B process.

Add
: This menu allows you to add nodes. Please see the next section for a discussion on the types of nodes that you can add, and what they do. Clicking this menu item is the same as pressing Space when the cursor is in the window

Node

Hide
: H - Hides your selected nodes. Just like vertices in a mesh.

Grouping
: Most importantly, this menu option allows you to create a user-defined group of nodes. This group can then be edited and added to the map. To create a group, select the nodes you want, and then Node → Make Group, or just use the keyboard shortcut CtrlG. Edit the name using the little input box in the group. Groups are easily identified by their green header and cool names you have picked for them.

Delete
: X - Deletes selected nodes.

Duplicate
: ⇧ ShiftD - Makes an Unlinked copy, with the same settings as the original.

Grab
: G - Moves the little nodes around according to your mouse, just like with meshes.

Duplicate - Faked you out
The new copy is placed **exactly over the old one**. But it isn't the connected one, so playing with the controls will do nothing to your images, even though it **looks** like it's connected with the little threads coming out of the node that is **underneath**. You have to move the duplicated node to reveal the connected node beneath it.

Grab - Reminder Only
Just like my mother-in-law, the menu item does not actually do anything; it's just there to remind you that you can press the G key when your cursor is in the window and actually accomplish something with your life (like rearranging nodes in the window).

Buttons

Material/Composite/Texture Selector

Nodes are grouped into two categories, based on what they operate on:

- to work with Material Nodes, click on the ball,
- to work with Compositing nodes, click on the overlaped pictures,
- to work with Texture nodes, click on the checker.

Use Nodes Button

This button tells the render engine to use the node map in computing the material color or rendering the final image, or not. If not, the map is ignored and the basic render of the material tabs or scene is accomplished.

Use Pinned Button

This button tells the render engine to use pinned node tree.

Go to Parent Button

This button allows you go to parent node tree.

Snap Button

Toggle snap mode for node in the Node Editor window.

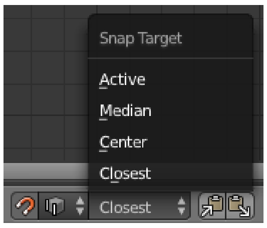
Snap Node Element selector
: This selector provide the follow node elements for snap:

  Grid (default)
  : Snap to grid of the Node Editor window.

  Node X
  : Snap to left/right node border.

  Node Y
  : Snap to top/bottom node border.

  Node X/Y
  : Snap to any node border.
  : Snapping to node border takes into account snap target:



Snap Target.

Snap Target
: Which part to snap onto the target
: Closest: Snap closest point onto target.
: Center: Snap center onto target.
: Median: Snap median onto target.
: Active: Snap active onto target.

Copy Nodes Button

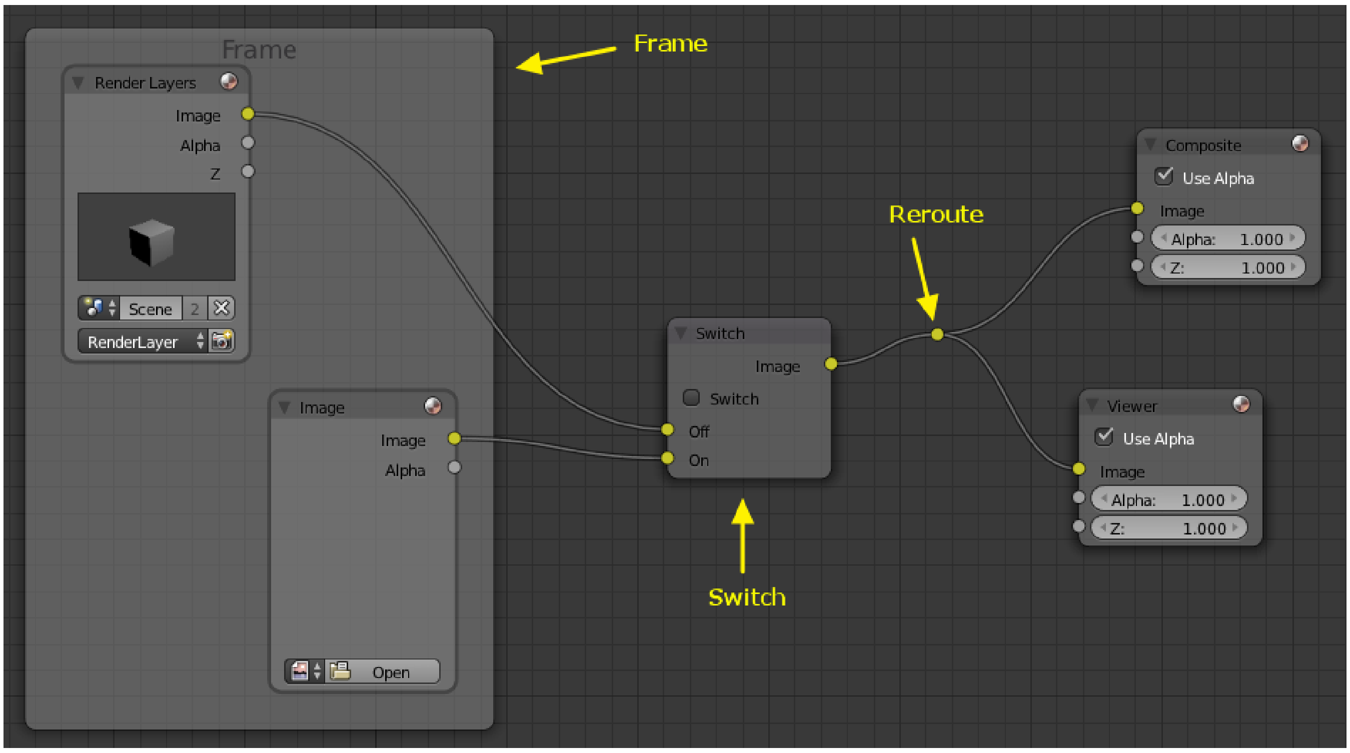
This button allows you copy selected nodes to the clipboard.

Paste Nodes Button

This button allows you paste nodes from the clipboard to the active node tree.

Layout Nodes

Layout nodes are designed for enhanced arrangement your nodes in Node Editor window. They are available from menu Add -> Layout.



Examples of the Layout Nodes.

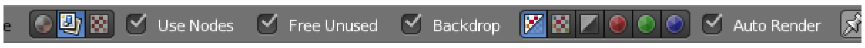
Blender provides the following layout nodes:

Frame
: This node is used for simple join a several nodes in the single place.

Reroute
: It's intended for branching threads with the purpose of optimization of Node Editor window's space.

Switch
: This node is applied for switching between color values. Available only for Compositing nodes.

Buttons for work with Compositing nodes



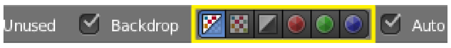
Buttons for work with Compositing nodes

Free Unused Button

This button frees up memory space when you have a very complex node map. Recommended.

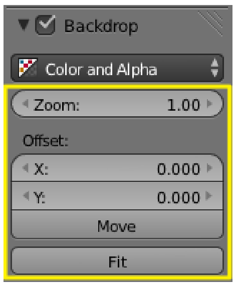
Backdrop

Use the active viewer node output as a backdrop. When enabled, additional settings appear in the Header and the Properties Panel:



Backdrop Channels.

Backdrop Channels
: Set the image to be displayed with Color, Color and Alpha, or just Alpha.



Options of Zoom and Offset of Backdrop.

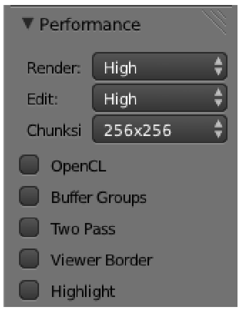
Zoom
: Sets how big the backdrop image is.

Offset
: Change the screen space position of the backdrop, or click the Move button, or shortcut Alt MMB to manually move it.

Auto Render

Re-render and composite changed layer when edits to the 3d scene are made.

Perfomance for Compositing Nodes in Node Editor



Perfomance for Compositing Nodes in Node Editor

Render
: Set quality when rendering in Node Editor.

Edit
: Set quality when editing in Node Editor

Chunksi
: Max size of a title (smaller values give better distribution of multiple threads, but more overhead).

OpenCL
: Enable GPU calculations when working in Node Editor.

Buffer Groups
: Enable buffering of group nodes.

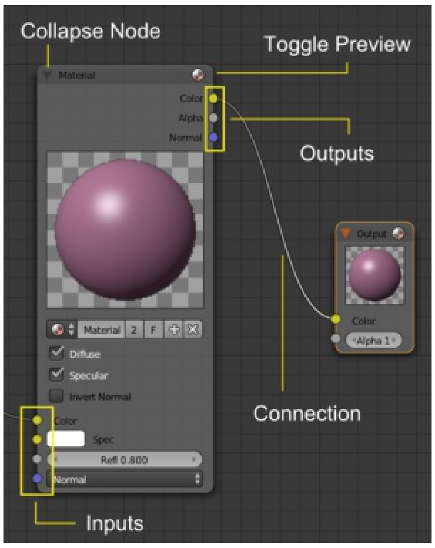
Two Pass
: Use two pass execution during editing: first calculate fast nodes, second pass calculate all nodes.

Viewer Border
: Use boundaries for viewer nodes and composite backdrop.

Highlight
: Highlight nodes that are being calculated.

Node Controls

This page explains the widgets to control a node.



Nodes main controls

Titlebar
: This contains the node's name, along with several different collapse buttons.

Input sockets
: The left side of a node has input sockets:

- *blue sockets* accept vectors.
- *yellowsockets* accept colors.
- *gray sockets* accept single values (like alpha).

Output sockets
: The right side of a node has output sockets:

- *blue sockets* produce vectors.
- *yellowsockets* produce colors.
- *gray sockets* produce single values (like alpha).

Image preview
: Inside the node there's an area to show the image preview being output by the node or the curves that control the node behavior (for example in a RGB node).

Buttons and menus
: Below the image preview there are buttons and menus to control the node behavior.

Threads
: A curved line shows a connection from an output socket to an input socket. The socket types must match.

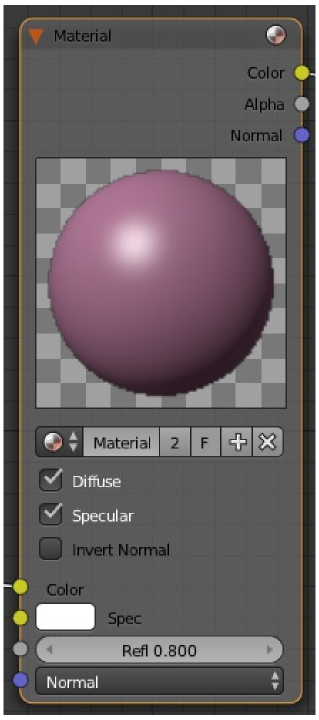
Connections associated with the active node are highlighted for better visibility.

Collapsing toggles

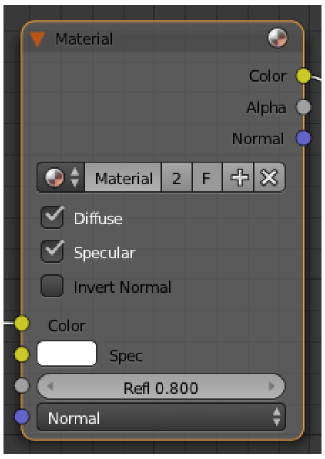
At the top of a node there are up to 4 visual controls for the node (*Top of a Node*). Clicking these controls influences how much information the node shows.

Node toggle (▼ ▶)
: The arrow on the left collapses/uncollapses the node.

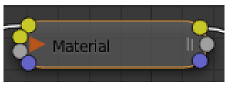
Preview image toggle
: The sphere button on the far right of the titlebar hides/unhides the preview image.



Full display



Preview hidden

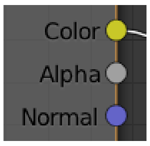


Node collapsed

Sizing the node

Fine Sizing of an individual node can also be accomplished somewhat by clicking LMB and dragging on the left or right edge of the node.

Sockets



Node Sockets.

Each Node in your node window will have "sockets" (often also referred to as "connectors") which are small colored circles to which input data and output data can be linked (*Node Sockets*).

The sockets on the left side of a node describe *inputs,* while the sockets on the right side are *outputs.*

For your convenience, nodes are *color-coded* according to the type of information they expect to send or receive. There are three colors:

● Yellow sockets
: Indicates that **color** information needs to be input or will be output from the node.

● Gray sockets
: Indicates values (**numeric**) information. It can either be a single numerical value or a so-called "value map." (You can think of a value map as a grayscale-map where the different amount of bright/dark reflects the value for each point.) If a single value is used as an input for a "value map" socket, all points of the map are set to this same value.
Common use: Alpha maps and value options for a node.

● Blue/Purple sockets
: Indicates **vector/coordinate/normal** information.

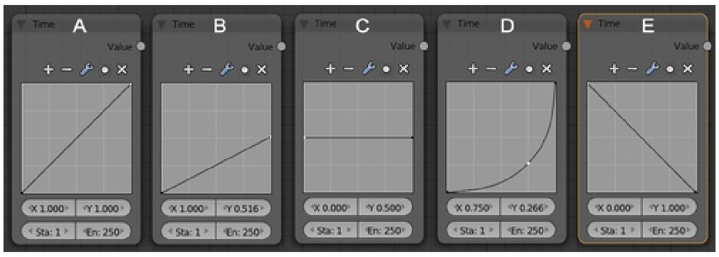
Between nodes, yellow must be linked to yellow, gray to gray, blue to blue, unless you use a *converter,* which we'll cover later on.

Next to the color in the node you will see the name of that socket. Though not always the case, you can think of the name of the socket as what the information is *intended* to be. But this is not necessarily what it *has* to be. For example, I can add a link from a gray socket titled Alpha to the material node's gray Reflection socket and still get a result, the key thing being that it's a "gray to gray" connection.

There are exceptions where you can mix yellow (i.e. a color image) and gray (*e.g.* grayscale) without converters. Blender normally places a converter if needed, so feel free to experiment with them. You can use the "Viewer" output nodes, as explained in the later sections, to see if/how it works.

Curves

Some nodes have a curve area that translates an input value to an output value. You can modify this curve shape by clicking on a control point and moving it, or adding a control point. Some examples are shown below:



Modifying a curve node.

Every curve starts out as a straight line with a slope of 1. The curve starts out with two tiny black control points at each end of the line. Clicking LMB on a control point selects it and it turns white.

Changing the curve affects how the output is generated. The input, X, usually proceeds linearly (at regular intervals) across the **bottom** axis. Go up until you hit the curve, and then over to the **right** to determine the Y output for that corresponding X. So, for the second example, as X goes from 0 to 1.0 across the bottom, Y varies from 0.0 to 0.5. In the third, as X goes from 0.0 to 1.0 across the bottom, Y stays constant at 0.5. So, in the picture above, these curves have the following effect on time: **A** don't affect, **B** slow down, **C** stop, **D** accelerate, and **E** reverse time.

The "Curves" widget is a built-in feature in Blender's UI, and can be used anywhere, provided the curve data itself is being delivered to this widget. Currently it is used in the Node Editor and in the UV Window.

This widget will map an input value horizontally and return the new value as indicated by the height of the curve.

Note: The fact that one of the points on the curve is "white" in each of these screenshots is *not* significant; it just means that it happened to be the point most-recently selected by your author when preparing this tutorial. What matters here is the shape of *the curve,* not the position (nor the color) of the control points that were used to define it.

RGB Curves

Multiple curves can be edited in a single widget. The typical use, RGB curves, has "Combined" result or "Color" ("C") as the first curve, and provides curves for the individual R, G, and B components. All four curves are active together; the "C" curve gets evaluated first.

Selecting curve points

- LMB always selects 1 point and deselects the rest.
- Hold ⇧ Shift while clicking to extend the selection or select fewer points.

Editing curves

- LMB click&drag on a point will move points.
- A LMB click on a curve will add a new point.
- Dragging a point exactly on top of another will merge them.
- Holding ⇧ Shift while dragging snaps to grid units.
- Ctrl LMB adds a point.
- Use the X icon to remove selected points.

Editing the view

The default view is locked to a 0.0-1.0 area. If clipping is set, which is the default, you cannot zoom out or drag the view. Disable clipping with the icon resembling a #.

- LMB click&drag outside of curve moves the view
- Use the + and - icons to zoom in or out.

Special tools

The wrench icon gives a menu with choices to reset a view, to define interpolation of points, or to reset the curve.

Using nodes

Adding Nodes

Nodes are added in two ways to the node editor window:

- By clicking the Add menu in the node editor toolbar and picking the type of node you want, or
- By clicking the ⇧ ShiftA -> Add and picking a node from the popup Add menu.

Arranging Nodes

In general, try to arrange your nodes within the window such that the image flows from left to right, top to bottom. Move a node by clicking on a benign area and dragging it around. The node can be clicked almost anywhere and dragged about; connections will reshape as a bezier curve as best as possible.

Connecting nodes

LMB -click and drag a socket: you will see a branch coming out of it: this is called a "thread".

Kepp dragging and connect the thread to an input socket of another node, then release the LMB .

In this case, a copy of each output is routed along a thread. However, only a single thread can be linked to an input socket.

Disconnecting nodes

To break a link between sockets Ctrl LMB -click in an empty areas near the thread you want to disconnect and drag: you will see a little cutter icon appearing at your mouse pointer. Move it over the thread itself, and release the LMB .

Duplicating a node

Click LMB or RMB on the desidered node, press ⇧ ShiftD and move the mouse away to see the duplicate of the selected node appeaing under the mouse pointer.

Gotcha!
When you duplicate a node, the new node will be positioned *exactly* on top of the node that was duplicated. If you leave it there (and it's quite easy to do so), you can **not** easily tell that there are *two* nodes there! When in doubt, grab a node and move it slightly to see if something's lurking underneath.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in the node editor, making your material or composite 'noodle' (node network) easier to work with. Grouping nodes also creates what are called NodeGroups (inside a .blend file) or NodeTrees (when appending).

Conceptually, "grouping" allows you to specify a *set* of nodes that you can treat as though it were "just one node." You can then re-use it one or more times in this or some other .blend file(s).

As an example: If you have created a material using nodes that you would like to use in another .blend file, you *could* simply append the material from one .blend file to another. However, what if you would like to create a new material, and use a branch from an existing material node network? You could re-create the branch. Or you could append the material to the new .blend file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different .blend files. What if you have created a “Depth of Field” composite node network and would like to use it in another .blend file? What if you wanted to apply exactly the same series of operations dozens of times? Here again, you *could* re-create the network, but this is not very efficient. A better method of re-use, for either material node branches or composite node networks, would be to create groups of nodes.

Once a group has been defined, it becomes an opaque object; a reusable software component. You can (if you choose) ignore exactly how it is *defined,* and simply use it (as many times as you like) for whatever it *does.* Groups can be made available through the Blender library and standard appending method.

Grouping Nodes

Panel: Node Editor

Menu: ⇧ ShiftA → Group → Make Group

To create a node group, in the node editor, select the nodes you want to include, then press CtrlG or ⇧ ShiftA » Group » Make Group. A node group will have a green title bar. All of the selected nodes will now be minimized and contained within the group node. Default naming for the node group is *NodeGroup, NodeGroup.001* etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one .blend file to another, Blender does not make a distinction between material node groups or composite node groups, so I recommend some naming convention that will allow you to easily distinguish between the two types. For example, name your material node branches *Mat_XXX,* and your composite node networks *Cmp_XXX.*

What not to include in your groups (all types of Node editors)

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not include**:

Source nodes
if you include a source node in your group, you'll end up having the source node appearing *twice:* once inside the group, and once outside the group in the new material node-network.

Examples of source nodes are: the *Material Node* (Material nodes editor) and the *Render Layers Node* (Composite Editor).

Output node
if you include an output node in the group, there won't be an output socket available *from* the group!

Examples of output nodes are: the *Output Node* (Material nodes editor) and the *Viewer Node* (Composite Editor).

Editing Node Groups

With a group node selected, pressing ⇆ Tab expands the node to a window frame, and the individual nodes within it are shown to you. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of your editor window. You will not be able to thread them to an outside node directly from them; you have to use the external sockets on the side of the Group node. To add or remove nodes from the group, you need to ungroup them.

Ungrouping Nodes

The AltG command destroys the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

Appending Node Groups

Once you have appended a NodeTree to your .blend file, you can make use of it in the node editor by pressing ⇧ ShiftA → Add → Group, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Types of Composite Nodes

This section is organized by type of nodes, which are grouped based on similar functions:

- Input - Adds something to the node map, such as an image or a value.
- Output - Displays the result in progress as a small image.
- Color - Manipulates the colors of an image.
- Vector - Manipulate the intensities and reflections of an image
- Filters - Process the image to enhance it, working on adjacent pixels.
- Convertors - Separate the image into its component video, or convert formats.
- Mattes - Generating mattes to mask off areas of an image.
- Distortion - Changing the shape of the image.
- Groups - User-defined groups of nodes.

Composite Input Nodes

Input nodes *produce* information from some source.

For instance, an input could be:

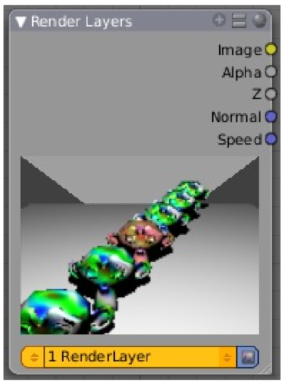
- taken directly from the active camera in a selected scene,
- from a JPG, PNG, etc. file as a static picture,
- a movie clip (such as an animation sequence or home movie), or
- just a color.

These nodes generate the information that feeds other nodes. As such, they have no input-connectors; only outputs.

Render Layers Node

Panel: Node Editor → Node Composition

Menu: ⇧ ShiftA → Input → Render Layers



Render Layers Node

This node is the starting place to getting a picture of your scene into the compositing node map.

This node inputs an image from a scene within your blend file. Select the scene and the active render layer from the yellow selection list at the bottom of the node. Blender uses the active camera for that scene to create an image of the objects specified in the RenderLayer.

The Image is input into the map, along with the following data:

- Alpha (transparency) mask

Depending on the Renderlayer passes that are enabled, other sockets are available. By default the Z is enabled:

- Z depth map (how far away each pixel is from the camera)

The example shows that two other passes are enabled:

- Normal vector set (how light bounces off the surface)
- Speed vector set (how fast an object is moving from one frame to the next)

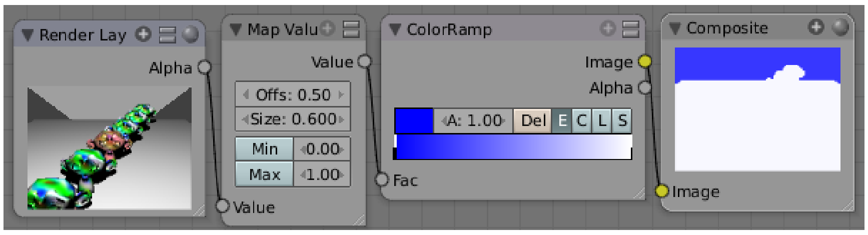
Use the re-render button (Small landscape icon - to the right of the Renderlayer name) to re-render the scene and refresh the image and map.

You may recall that a .blend file may contain many scenes. The Renderlayer node can pick up the scene info from any available scene by selecting the scene from the left-hand selector. If that *other* scene also uses the compositor and/or sequencer, you should note that the scene information taken is the raw information (pre-compositing and pre-sequencing). If you wish to use composited information from another scene, you will have to render that scene to a multilayer OpenEXR frameset as an intermediate file store, and then use the Image input node instead.

Using the Alpha Socket

Using the Alpha output socket is crucial in overlaying images on top of one another and letting a background image "show through" the image in front of it.

In a Blender scene, your objects are floating out there in virtual space. While some objects are in front of one another (Z depth), there is no ultimate background. Your world settings can give you the illusion of a horizon, but it's just that: an illusion. Further, some objects are semi-transparent; this is called having an Alpha value. A semi-transparent object allows light (and any background image) to pass through it to the camera. When you render an image, Blender puts out, in addition to a pretty image, a map of what solid objects actually are there, and where infinity is, and a map of the alpha values for semi-transparent objects. You can see this map by mapping it to a blue screen:



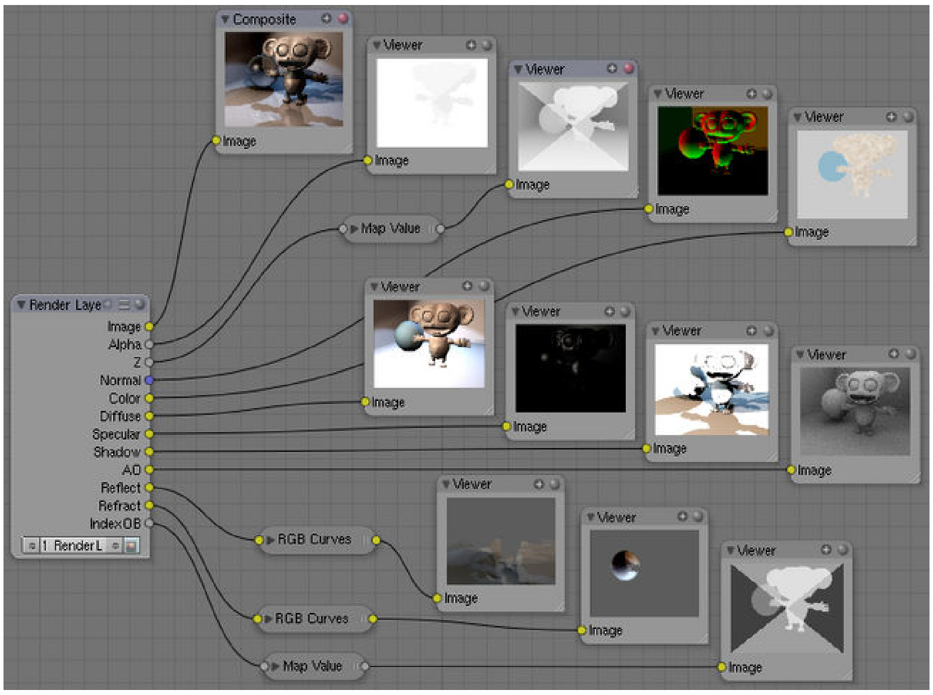
Viewing the Alpha values

In the little node map above, we have connected the Alpha output socket of the RenderLayer node to a Map Value node (explained later, but basically this node takes a set of values and maps them to something we can use). The Color Ramp node (also explained later in detail) takes each value and maps it to a color that we can see with our eyes. Finally, the output of the Color Ramp is output to a Composite viewer to show you, our dear reader, a picture of the Alpha values. Notice that we have set up the map so that things that are perfectly solid (opaque) are white, and things that are perfectly transparent (or where there is nothing) are blue.

Optional Sockets

For any of the optional sockets to appear on the node, you MUST have the corresponding pass enabled. In order for the output socket on the RenderLayer node to show, that pass must be enabled in the RenderLayer panel in the Buttons window. For example, in order to be able to have the Shadow socket show up on the RenderLayer input node, you must have the "Shad" button enabled in the Buttons window, Scene Render buttons, Renderlayer panel. See the RenderLayer tab (Buttons window, Output frame, Render Layers tab, Passes selector buttons) for Blender to put out the values corresponding to the socket.

For a simple scene, a monkey and her bouncy ball, the following picture expertly provides a great example of what each pass looks like:



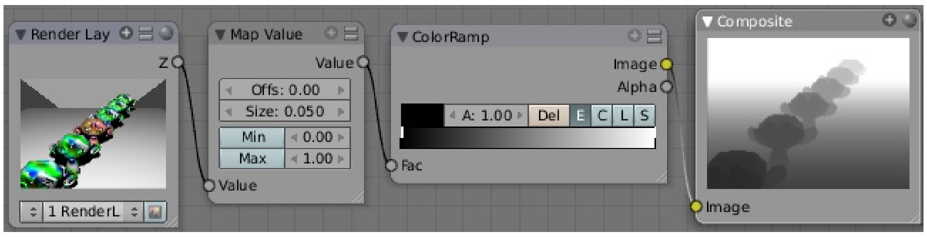
The available sockets are:

- Z: distance away from the camera, in Blender Units
- Normal (Nor): How the color is affected by light coming from the side
- UV: how the image is distorted by the UV mapping
- Speed (Vec): How fast the object is moving, and in what direction
- Color (Col): the RGB values that color the image that you see
- Diffuse: the softening of colors as they diffuse through the materials
- Specular: the degree of shininess added to colors as they shine in the light
- Shadow: shadows cast by objects onto other objects
- AO: how the colors are affected by Ambient Occlusion in the world
- Reflect (Ref): for mirror type objects, the colors they reflect and are thus not part of their basic material
- Refract: how colors are bent by passing through transparent objects
- Radio (Radiosity): colors that are emitted by other objects and cast onto the scene
- IndexOB: a numeric ordinal (index) of each object in the scene, as seen by the camera.

Using the Z value Socket

Using the Z output socket is crucial in producing realistic images, since items farther away are blurrier (but more on that later).

Imagine a camera hovering over an X-Y plane. When looking through the camera at the plane, Y is up/down and X is left/right, just like when you are looking at a graph. The camera is up in the air though, so it has a Z value from the X-Y plane, and, from the perspective of the camera, the plane, in fact all the objects that the camera can see, have a Z value as a distance that they are away from it. In addition to the pretty colors of an image, a RenderLayer input node also generates a Z value map. This map is a whole bunch of numbers that specify how far away each pixel in the image is away from the camera. You can see this map by translating it into colors, or shades of gray:



Viewing the Z values

In the little node map above, we have connected the Z output socket of the RenderLayer node to a Map Value node (explained later). This node takes a set of values and maps them to something we can use. The Color Ramp node (also explained later in detail) takes each value and maps it to a shade of gray that we can see with our eyes. Finally, the output of the colorramp is output to a Composite viewer to show you, our dear reader, a picture of the Z values. Notice that we have set up the Map Value node so that things closer to the camera appear blacker (think: black is 0, less Z means a smaller number) and pixels/items farther away have an increasing Z distance and therefore get whiter. We chose a Size value of 0.05 to see Z values ranging from 0 to 20 (20 is 1/0.05).

Using the Speed Socket

Even though things may be animated in our scene, a single image or frame from the animation does not portray any motion; the image from the frame is simply where things are at that particular time. However, from the Render Layers node, Blender puts out a vector set that says how particular pixels are moving, or will move, to the next frame. You use this socket to create a blurring effect. Find out more by clicking here.

Image node

Panel: Node Editor → Node Composition

Menu: ⇧ ShiftA → Input → Image

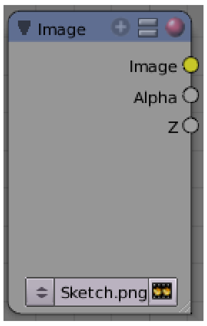


Image node

The Image node injects any image format that is supported by Blender. Besides inputting the actual image, this node can also input Alpha and depth (Z) values if the image has them. If the image is a MultiLayer format, all saved render passes are input. Use this node to input:

- A single image from a file (such as a JPG picture)
- Part or all of an animation sequence (such as the 30th to 60th frame)
- Part or all of a movie clip (such as an AVI file)
- the image that is currently in the UV/Image Editor (and possibly being painted)
- an image that was loaded in the UV/Image Editor

Animated image sequences or video files can also be used. See Animations below.

To select an image file or generated image from the UV/Image Editor, click on the small arrow selector button to the left of the name and pick an existing image (e.g. loaded in the UV editor or elsewhere) or click on LOAD NEW to select a file from your hard disk via a file-browser. These images can be e.g. previously rendered images, matte paintings, a picture of your cat, whatever. Blender really doesn't care.

If the image is part of a sequence, manually click the Image Type selector to the right of the name, and select *Sequence*. Additional controls will allow you to define how much of the sequence to pull in (see Animations below). If the file is a video file, these controls will automatically appear.

Image Channels

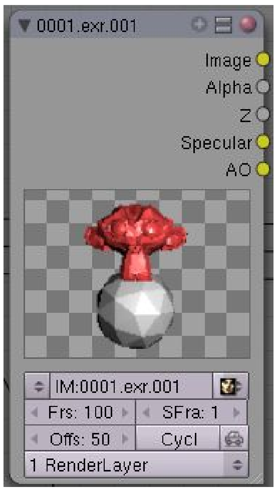
When the image is loaded, the available channels will be shown as sockets on the node. As a minimum, the Image, Alpha, and Z channels are made available. The picture may or may not have an alpha (transparency) and/or Z (depth) channel, depending on the format. If the image format does not support A and/or Z, default values are supplied (1.0 for A, 0.0 for Z).

- Alpha/Transparency Channel
  - If a transparency channel is detected, the Alpha output socket will supply it.
  - If it does not have an Alpha channel (e.g. JPG images), Blender will supply one, setting the whole image to completely opaque (an Alpha of 1.00, which will show in a Viewer node as white - if connected to the Image input socket).

- Z/depth Channel
    - If a Z (depth) channel is detected, the Z output socket will supply it.
    - If it does not have a Z channel (e.g. JPG or PNG images), Blender will supply one, setting the whole image to be at the camera (a depth of 0.00). To view the Z-depth channel, use the Map Value to ColorRamp noodle given above in the Render Layer input node, in the Render Layer input node.

Formats
Blender supports many image formats. Currently only the OpenEXR image format stores RGB (color), A (alpha), and Z (depth) buffer information in a single file, if enabled.

Saving/Retrieving Render Passes



Blender can save the individual Render Layers and specific passes in a MultiLayer file format, which is an extension of the OpenEXR format. In this example, we are reading in frames 50 to 100 of a RenderLayer that were generated some time ago. The passes that were saved were the Image, Alpha, Z, Specular and AO passes.

To create a MultiLayer image set when initially rendering, simply disable Do Composite, set your Format to MultiLayer, enable the Render Layer passes you wish to save over the desired frame range, and Animate. Then, in Blender, enable Compositing Nodes and Do Composite, and use the Image input node to read in the EXR file. When you do, you will see each of the saved passes available as sockets for you to use in your compositing noodle.

Image Size

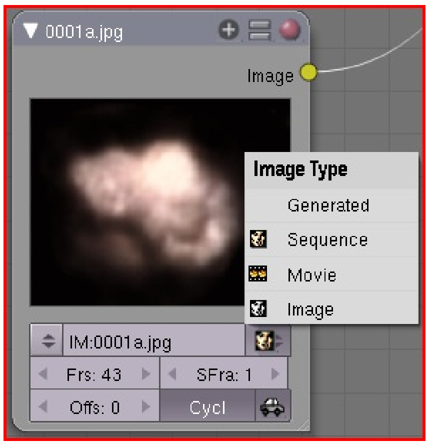
Size matters - Pay attention to image resolution and color depth when mixing and matching images. Aliasing (rough edges), color *flatness*, or distorted images can all be traced to mixing inappropriate resolutions and color depths.

The compositor can mix images with any size, and will only perform operations on pixels where images have an overlap. When nodes receive inputs with differently sized Images, these rules apply:

- The first/top Image input socket defines the output size.
- The composite is centered by default, unless a translation has been assigned to a buffer using a Translate node.

So each node in a composite can operate on different sized images, as defined by its inputs. Only the Composite output node has a fixed size, as defined by the Scene buttons (Format Panel - F10). The Viewer node always shows the size from its input, but when not linked (or linked to a value) it shows a small 320x256 pixel image.

Animations



To use image sequences or movies within your composition, press the face or little film strip button located to the right of the selector. As you click, a pop-up will offer you four choices:

1. Generated -
2. Sequence - a sequence of frames, each frame in a separate file.
3. Movie - a sequence of frames packed into a single .avi or .mov file
4. Image - a single frame or still image in a file

A Movie or Image can be named anything, but a Sequence must have a digit sequence somewhere in its filename, for example fire0001set.jpg, fire0002set.jpg, fire0003set.jpg and so on. The number indicates the frame.

If a Sequence or Movie is selected, an additional set of controls will appear that allows you to select part or all of the sequence. Use these controls to specify which frames, out of the original sequence, that you want to introduce into the animation you are about to render. You can start at the beginning and only use the beginning, or even pick out a set of frames from the middle of an existing animation.

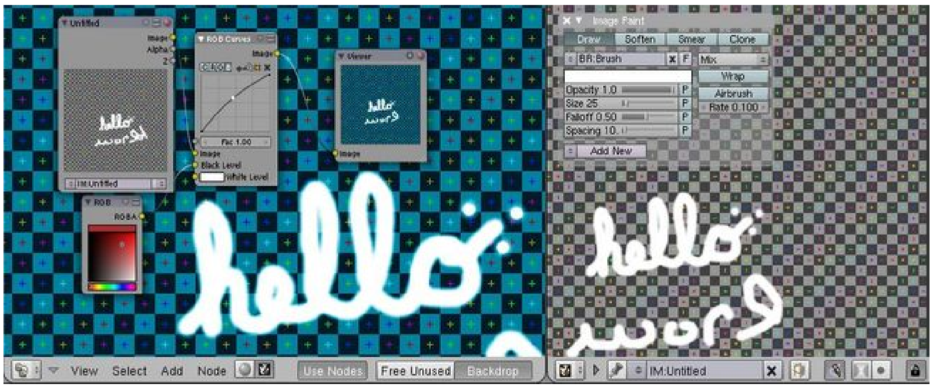
The Frs number button is the number of frames in the sequence that you want to show. For example, if you want to show 2 seconds of the animation, and are running 30 fps, you would put 60 here.

The SFra number button sets the start frame of the animation; namely, at what point in the animation that you *are going to render* do you want this sequence to start playing. For example, if you want to introduce this clip ten seconds into the composite output, you would put 300 here (at 30 fps).

The First number button sets the first number in the animated sequence name. For example, if your images were called "credits-0001.png", "credits-0002.png" through "credits-0300.png" and you wanted to start picking up with frame 20, you'd put 20 here.

To have the movie/sequence start over and repeat when it is done, press the Cyclic button. For example, if you were compositing a fan into a room, and the fan animation lasted 30 frames, the animation would start over at frame 31, 61, 91, and so on, continuously looping. As you scrub from frame to frame, to see the actual video frame used for the current frame of animation, press the auto button to the right of the Cyclic button.

Generated Images



Using the Nodes to modify a painting in progress in the UV/Image window

Blender features Texture Paint which works in the UV/Image Editor, that allows you to paint on the fly, and the image is kept in memory or saved. If sync lock is enabled (the lock icon in the header), changes are broadcast throughout Blender as soon as you lift the mouse button. One of the places that the image can go is to the Image Input node. The example shows a painting session going on in the right-hand UV/Image Editor window for the painting "Untitled". Create this image via Image->New in the UV/Image Editor. Refer to the texture paint section of the user maual for more info on using Texture Paint.

In the left-hand window, the Image input node was used to select that "Untitled" image. Notice that the Image type icon is blank, indicating that it is pulling in a Generated image. That image is colorized by the noodle, with the result used as a backdrop in the Node Editor Window.

Using this setup and the Generated Image type is like painting and post-processing as you continue painting. Changes to either the painting or the post-pro noodle are dynamic and real-time.

Notes

No Frame Stretching or Compression: If the input animation (avi or frame set) was encoded at a frame rate that is *different* from your current settings, the resultant animation will appear to run faster or slower. Blender Nodes do not adjust input video frame rates. Use the scale control inside the Video Sequence Editor to stretch or compress video to the desired speed, and input it here. You can incorporate "Slow-Mo" into your video. To do so, ANIMate a video segment at 60 frames per second, and input it via this node, using Render settings that have an animation frame rate of the normal 30 fps; the resulting video will be played at half speed. Do the opposite to mimic Flash running around at hyperspeed.

AVI (Audio Video Interlaced) files are encoded and often compressed using a routine called a *Codec*. You must have a codec installed on your machine and available to Blender that understands and is able to read the file, in order for Blender to be able to de-code and extract frames from the file. If you get the error message **FFMPEG or unsupported video format** when trying to load the file, you need to get a Codec that understands the video file. Contact the author of the file and find out how it was encoded. An outside package, such as VirtualDub, might help you track this information down. Codecs are supplied by video device manufacturers, Microsoft, DivX, and Xvid, among others, and can often be downloaded from their web sites for free.

Splicing Video Sequences using Nodes

The above animation controls, coupled with a little mixing, is all you need to splice video sequences together. There are many kinds of splices:

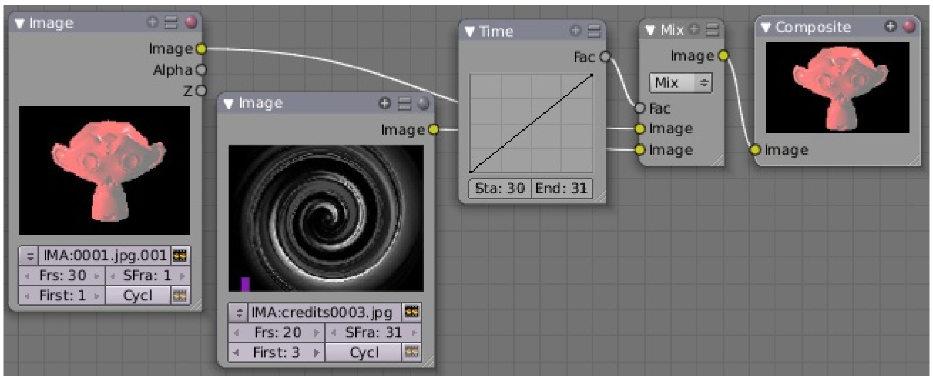
- Cut Splice - literally the ends of the footage are just stuck together
- Fade In - The scene fades in, usually from black
- Fade Out - The scene fades out, usually to black
- Mix - Toward the end of one scene, the images from the next scene meld in as the first scene fades
- Winking and Blinking - fading one cut out while the other fades in, partially or totally through black
- Bumps and Wipes - one cut bumps the other one out of frame, or wipes over it (like from the top left corner down)

Cut Splicing using Nodes

In the example noodle below, we have two pieces of footage that we want to cut splice together.

- Magic Monkey - named 0001.png through 0030.png
- Credits - named credits0001.png through credits0030.png

The editor has reviewed the Credits and thought the first two frames could be thrown away (onto the cutting room floor, as they say) along with the last 8, leaving 20 frames from the total shot. Not shown in this image, but crucial, is that in the Output panel, we set our render output filename to "Monkey-Credits-", and our Animation start and end frames to 1 and 50 (30 from the Monkey, 20 from the credits). Notice the Time node; it tells the Mix node to use the top image until frame 30, and then, at frame 31, changes the Mix factor to 1, which means to use the bottom set of images.



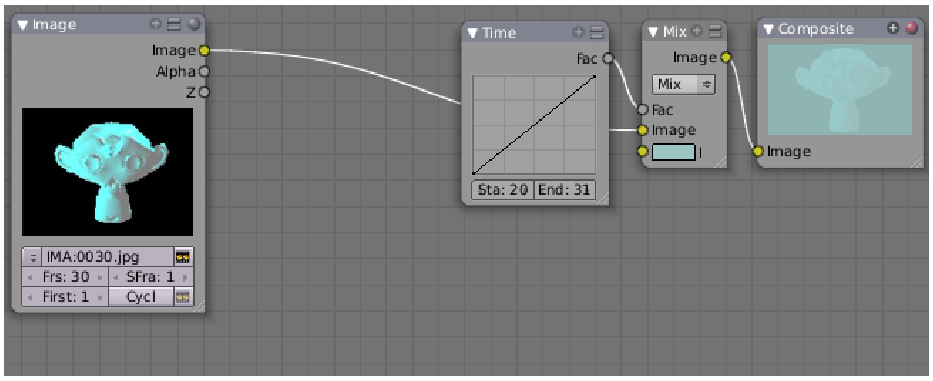
Cut Splice using Nodes

Upon pressing the ANIM button, Blender will composite the animation. If you specified an image format for output, for example, PNG, Blender will create 50 files, named "Monkey-Credits-0001.png" through "Monkey-Credits-0050.png". If you specified a movie format as output, such as AVI-JPEG, then Blender will create only one file, "Monkey-Credits-.avi", containing all 50 frames.

Use cut scenes for rapid-fire transition, conveying a sense of energy and excitement, and to pack in a lot of action in a short time. Try to avoid cutting from a dark scene to a light one, because it's hard on the eyes. It is very emotionally contrasting, and sometimes humorous and ironic, to cut from a very active actor in one scene to a very still actor in another scene, a la old Road Runner and Coyote scenes.

Fade Splicing using Nodes

In the previous topic, we saw how to cut from one sequence to another. To fade in or out, we simply replace one set of images with a flat color, and expand the Time frame for the splice. In the image below, beginning at frame 20, we start fading **out** to cyan:



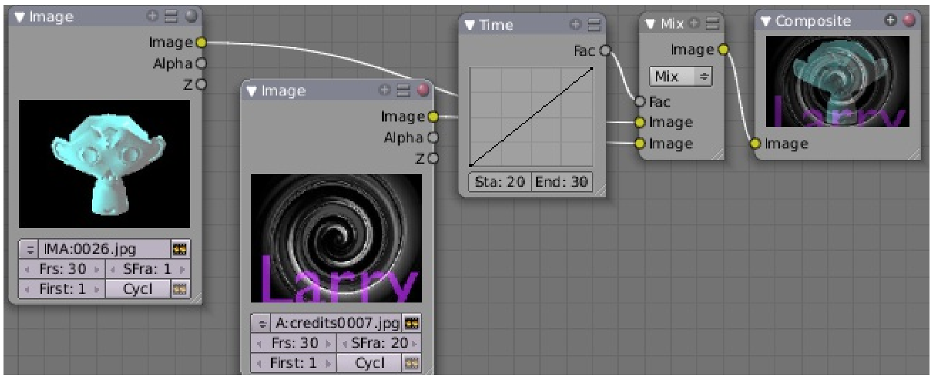
Fading Out using Nodes

Cyan was chosen because that is the color of the Monkey at that time, but you can just as easily choose any color. The image below shows frame 30, when we have almost faded completely.

To fade **in**, change the Mix node and plug the image sequence into the bottom socket, and specify a flat color for the top socket.

Mix Splice using Nodes

To mix, or crossover, from one scene to the next, start feeding the second scene in while the first is mixing out. The noodle below shows frame 25 of a mix crossover special effect to transition from one scene to the next, beginning at frame 20 with the transition completed by frame 30. Action continues in the first scene as it fades out and is mixed with action that starts in the second scene.

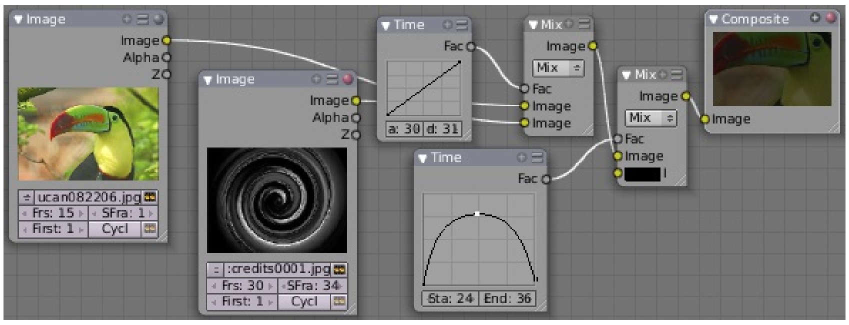


Mix Splice using Nodes

Use this effect to convey similarities between the two scenes. For example, Scene 1 is the robber walking down the street, ending with the camera focusing in on his feet. Scene 2 is a cop walking down the street after him, starting with his feet and working its way up to reveal that the cop is following the robber.

Wink Splice using Nodes

A Wink is just like blinking your eyes; one scene fades to black and the other fades in. To use Blender to get this effect, build on the Cut and Fade splices discussed above to yield:



A Wink using Nodes

In the above example, showing frame 27, we have adjusted some parameters to show you the power of Blender and how to use its Nodes to achieve just the blended crossover effect you desire:

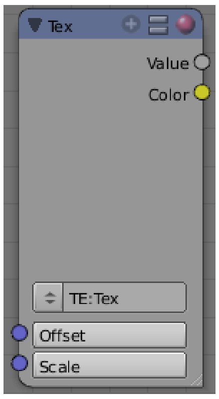
- Postfeed: Even though there were only 15 frames of animation in the Toucan strip, the cutover (top Time node) does not occur until frame 30. Blender continues to put out the last frame of an animation, *automatically extending it for you*, for frames out of the strip's range.
- Prefeed: Even though the swirl does not start playing until frame 34, Blender supplies the first frame of it for Frames 31 through 33. In fact, it supplies this image all the way back to frame 1.
- Partial Fade: Notice the second 'wink' Time node. Like a real wink, it does not totally fade to black; only about 75%. When transitioning between scenes where you want some visual carryover, use this effect because there is not a break in perceptual sequence.

Multiple Feeds
The above examples call out two feeds, but by replicating the Input, Time and Mix nodes, you can have multiple feeds at any one time; just set the Time node to tell the Mixer when to cut over to using it.

Texture Node

Panel: Node Editor → Node Composition

Menu: ⇧ ShiftA → Input → Texture



Texture node

The Texture node makes 3D textures available to the compositor.

The Texture node makes 3D textures available to the compositor. A texture, from the list of textures available in the current blend file, is selected and introduced through the value and/or color socket.

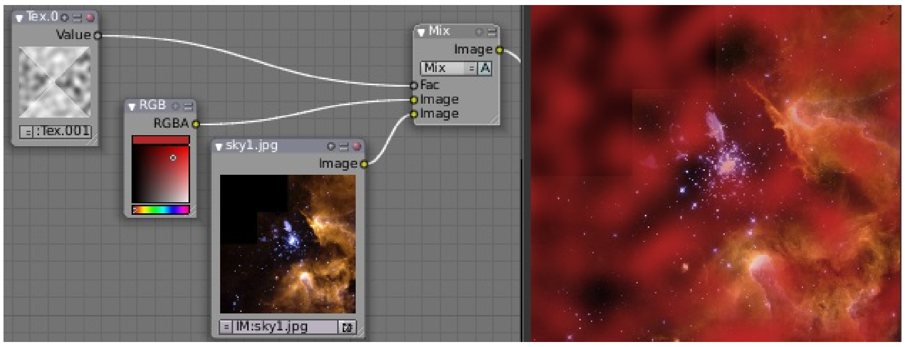
Note
Please read up on the Blender Library system for help on importing and linking to textures in other blender files.

Note
You cannot edit the textures themselves in the node window. To use this node, create and edit the texture in the normal texture buttons, then select the texture from the menu button on the node.

You can change the Offset and a Scale (which is called Offs XYZ and Size XYZ in the Materials Texture Map Input panel) for the texture by clicking on the label and setting the sliders, thus affecting how the texture is applied to the image. For animation, note that this is a vector input socket, because the XYZ values are needed.

Texture nodes can output a straight black-and-white Value image (don't mistake this for alpha) and an image (Color).

Example



In the example above, we want to simulate some red plasma gas out there in space. So, we fog up an image taken from the Hubble telecscope of Orion and take the ever-so-useful Cloud texture and use it to mix in red with the image.

Value node

Panel: Node Editor → Node Composition

Menu: ⇧ ShiftA → Input → Value

The Value node has no inputs; it just outputs a numerical value (floating point spanning 0.00 to 1.00) currently entered in the NumButton displayed in its controls selection.

Use this node to supply a constant, fixed value to other nodes' value or factor input sockets.

RGB node

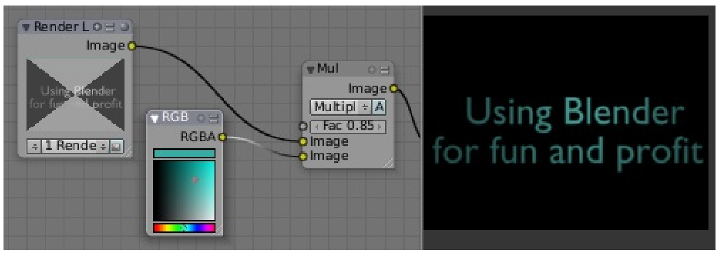
Panel: Node Editor → Node Composition

Menu: ⇧ ShiftA → Input → RGB

The RGB node has no inputs. It just outputs the Color currently selected in its controls section; a sample of it is shown in the top box. In the example to the right, a gray color with a tinge of red is slected.

To change the brightness and saturation of the color, LMB click anywhere within the square gradient. The current saturation is shown as a little circle within the gradient. To change the color itself, click anwhere along the rainbow Color Ramp.

Example

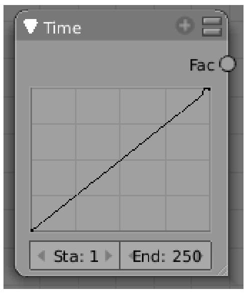


In this example, our corporate color is teal, but the bozo who made the presentation forgot. So, we multiply his lame black and white image with our corporate color to save him from embarassment in front of the boss when he gives his boring presentation.

Time node

Panel: Node Editor → Node Composition

Menu: ⇧ ShiftA → Input → Time



Time node

The Time node generates a factor value (from 0.00 to 1.00) (that changes according to the curve drawn) as time progresses through your movie (frames).

The Start and End NumButtons specify the range of time the values should be output along, and this range becomes the X-axis of the graph. The curve defines the Y-value and hence the factor that is output. In the example to the right, since the timespan is 250 frames and the line is straight from corner to corner, 0.50 would be output at frame 125, and 0.75 will be output at frame 187.

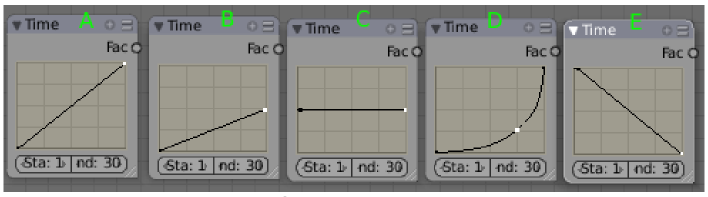
Note on output values
The Map Value node can be used to map the output to a more appropriate value. With some time curves, it is possible that the Time node may output a number larger than one or less than zero. To be safe, use the Min/Max clamping function of the Map Value node to limit output.

You can reverse time (unfortunately, only in Blender and not in the real world) by specifying a Start frame greater than the End frame. The net effect of doing so is to flip the curve around. Warning: doing so is easily overlooked in your node map and can be very confusing (like meeting your mother when she was/is your age in "Back to the Future").

Time is Relative
In Blender, time is measured in frames. The actual duration of a time span depends on how fast those frames whiz by (frame rate). You set the frame rate in your animation settings (Scene Context F10). Common settings range from 5 seconds per frame for slideshows (0.2 fps), to 30 fps for US movies.

Time Node Examples

In the picture below, over the course of a second of time (30 frames), the following time controls are made:



A) No Effect **B**) Slow Down **C**) Freeze **D**) Accelerate **E**) Reverse

Common uses for this include a "fade to black", wherein the accelerate time curve (typically exponentially-shaped) feeds a mix value that mixes a constant black color in, so that the blackness accelerates and eventually darkens the image to total black. Other good uses include an increasing soften (blur-out or -in) effect, or fade-in a background or foreground, instead of just jumping things into or out of the scene.

You can even imagine hooking up one blur to a background renderlayer, another inverted blur to a foreground renderlayer, and time-feeding both. This node group would simulate someone focusing the camera lens.

Examples and suggestions

As your imagination runs wild, consider a few ideas that came to me just now on my couch: mixing a clouds texture with a time input to fog up a piece of glass or show spray paint building up on a wall. Consider mixing red and the soften with time (decreasing output) to show what someone sees when waking up from a hard hit on the head. Mix HSV input with a starfield image with time (decreasing output) to show what we might see someday as we accelerate our starship and experience red-shift.

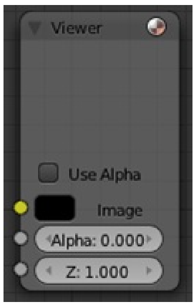
As a user, you should know that we have arrived at the point where there are many ways to do the same thing in Blender. For example, an old way to make a slide show using Blender, you created multiple image textures, one image for each slide, and assigned them as texture channels to the material for the screen, then created a screen (plane) that filled the cameral view. Using a material ipo, you would adjust the Color influence of each channel at different frames, fading one in as the previous slide faded out. Whew! Rearranging slide and changing the timing was clunky but doable by moving the IPO keys. The *Node* way is to create an image input, one for each slide image. Using the Image input and Time nodes connected to an AlphaOver mixer is much simpler, clearer, and easier to maintain.

Composite Output Nodes

At any point, you may want to see or save the working image in progress, especially right after some operation by a node. Simply create another thread from the image output socket of the node to an Output node to see a mini-picture.

Only one Viewer and one Composite Node is active, which is indicated with a red sphere icon in the Node header. Clicking on Viewer Nodes makes them active. The active Composite Node is always the first, and you should only use one anyway.

Viewer



Viewer node

The Viewer node is a temporary, in-process viewer. Plug it in wherever you would like to see an image or value-map in your node-tree.

LMB click on the image to update it, if it wasn't done automatically. You can use as many of these as you would like. It is possible to automatically plug a Viewer node to any other node by pressing ⇧ ShiftCtrl LMB on it.

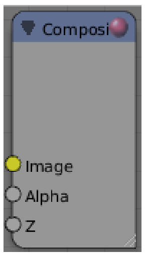
Using the UV/Image Editor Window

The Viewer node allows results to be displayed in the UV/Image Editor. The image is facilitated by selecting the IM:Viewer Node on the window's header. The UV/Image Editor will display the image from the currently selected viewer node.

To save the image being viewed, use the Image->Save As menu to save the image in a file.

The UV/Image Editor also has three additional options in its header to view Images with or without Alpha, or to view the Alpha or Z itself. Holding LMB in the Image display allows you to sample the values.

Composite



Composite node

The Composite node is where the actual output from the compositor is connected to the renderer. Connecting a node to the Composite node will output the result of that node's full tree to the Renderer; leaving this node unconnected will result in a blank image. This node is updated after each render, but also if you change things in your node-tree (provided at least one finished input node is connected).

You can connect three channels: the actual RGBA image, the Alpha image, and the Z (depth) image. You should only have one Composite node in your map so that only one final image is rendered when the Compositing button is pressed on the Render Options Post-Processing panel. Otherwise, unpredictable results may occur.

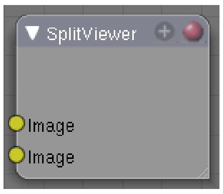
Saving your Composite Image

The RENDER button renders a single frame or image. Save your image using F3 or the File->Save Image menu. The image will be saved using the image format settings on the Render panel.

To save a sequence of images, for example, if you input a movie clip or used a Time node with each frame in its own file, use the ANIM button and its settings. If you might want to later overlay them, be sure to use an image format that supports an Alpha channel (such as PNG). If you might want to later arrange them front to back or create a depth of field effect, use a format that supports a Z-depth channel (such as EXR).

To save a composition as a movie clip (all frames in a single file), use an AVI or Quicktime format, and use the ANIM button and its settings.

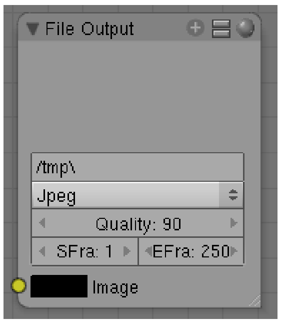
SplitViewer Node



SplitViewer node

The SplitViewer node takes two images and displays one half of each on each side (top socket on the right half, bottom socket input on the left). Use this node for making side-by-side comparisons of two renderings/images, perhaps from different renderlayers or from different scenes. When transitioning between scenes, you want to be sure the stop action is seamless; use this node to compare the end of one scene with the beginning of another to ensure they align.

File Output Node



File Output node

This node puts out an RGBA image, in the format selected, for each frame range specified, to the filename entered, as part of a frameset sequence. This means that the name of the file will be the name you enter plus a numeric frame number, plus the filename extension (based on format). Based on the format you choose, various quality/compression options may be shown.

To support subsequent arrangement and layering of images, the node can supply a Z-depth map. However, please note that only the OpenEXR image formats save the Z information.

The image is saved whenever Blender feels like it. Just kidding; whenever you press the Render button, the current frame image is saved. When you press the Anim button, the frameset sequence (specified in the Start and End frame) is saved.

This node saves you from doing (or forgetting to do) the Save Image after a render; the image is saved automagically for you. In addition, since this node can be hooked in anywhere in the noodle, you can save intermediate images automatically. Neat, huh?

Filespecs
As with all filename entries, use // at the beginning of the field to shorthand reference the current directory of the .blend file. You can also use the .. breadcrumb to go up a directory.

Levels Node

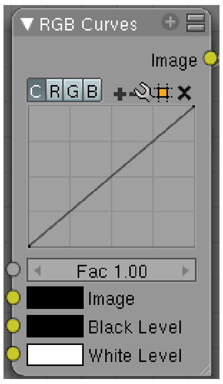
The Levels Node takes an image as an input, and can output a 1D value based on the levels of an image. It can read the input's Combined RGB, Red, Green, Blue, or Luminance channels.

It can output a Mean value, or average of values, or a Standard deviation, which measures the diversity of values.

Composite Color Nodes

These nodes play with the colors in the image. They adjust the image's color intensity, adjust contrast and intensity, and, most importantly, mix two images together by color, transparency, or distance.

RGB Curves Node

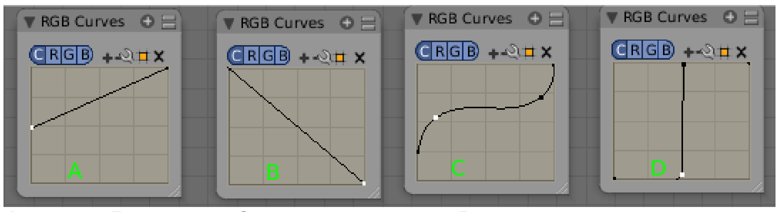


RGB Curves node

For each color component channel (RGB) or the composite (C), this node allows you to define a bezier curve that varies the input (across the bottom, or x-axis) to produce an output value (the y-axis). By default, it is a straight line with a constant slope, so that .5 along the x-axis results in a .5 y-axis output. Click and drag along the curve to create a control point and to change the curve's shape. Use the X to delete the selected (white) point.

Clicking on each C R G B component displays the curve for that channel. For example, making the composite curve flatter (by clicking and dragging the left-hand point of the curve up) means that a little amount of color will result in a lot more color (a higher Y value). Effectively, this bolsters the faint details while reducing overall contrast. You can also set a curve just for the red, and for example, set the curve so that a little red does not show at all, but a lot of red does.

Here are some common curves you can use to achieve desired effects:



A) Lighten **B**) Negative **C**) Decrease Contrast **D**) Posterize

Options

Fac
: How much the node should factor in its settings and affect the output.

Black Level
: Defines the input color that is mapped to black. Default is black, which does not change the image.

White Level
: Defines the input color that is mapped to white. Default is white, which does not change the image.

The levels work exactly like the ones in the image viewer. Input colors are scaled linearly to match black/white levels.

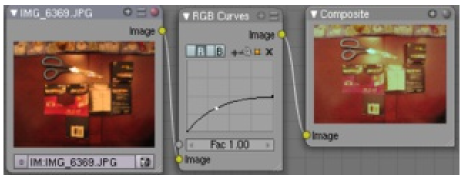
To define the levels, either use LMB on the color patch to bring up the color selection widget or connect some RGBA input to the sockets.

To only affect the value/contrast (not hue) of the output, set the levels to shades of gray. This is equivalent to setting a linear curve for C.

If you set any level to a color with a saturation greater than 0, the output colors will change accordingly, allowing for basic color correction or effects. This is equivalent to setting linear curves for R, G and B.

Examples

Color correction using Curves

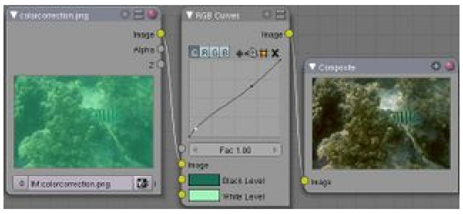


Color correction with curves

In this example, the image has way too much red in it, so we run it through an RGB node and reduce the Red channel by about half.

We added a middle dot so we could make the line into a sideways exponential curve. This kind of curve evens out the amount of a color in an image as it reaches saturation. Also, read on for examples of the Darken and Contrast Enhancement curves.

Color correction using Black/White Levels



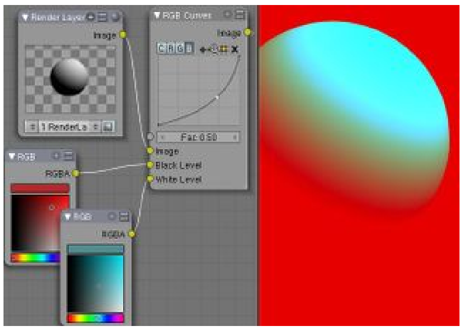
Color correction with Black/White Levels

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it's best to bring up an image viewer window showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming in to pixel level if necessary. The result can be fine-tuned with the R,G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side-effect of setting Black and White Levels.

Effects



Changing colors

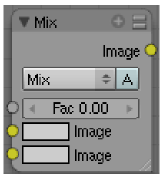
Curves and Black/White Levels can also be used to completely change the colors of an image.

Note that e.g. setting Black Level to red and White Level to blue does not simply substitute black with red and white with blue as the example image might suggest. Levels do color scaling, not substitution, but depending on the settings they can result in the described color substitution.

(What really happens when setting Black Level to pure red and White Level to pure blue is that the red channel gets inverted, green gets reduced to zero and blue remains unchanged.)

Because of this the results of setting arbitrary Black/White Levels or RGB curves is hard to predict, but can be fun to play with.

Mix Node



This node mixes a base image (threaded to the top socket) together with a second image (bottom socket) by working on the individual and corresponding pixels in the two images or surfaces. The way the output image is produced is selected in the drop-down menu. The size (output resolution) of the image produced by the mix node is the size of the base image. The alpha and Z channels are mixed as well.

Not one, not two, but count 'em, *sixteen* mixing choices include:

Mix
: The background pixel is covered by the foreground using alpha values.

Add
: The pixels are added together. *Fac* controls how much of the second socket to add in. Gives a bright result.
: The "opposite" to Subtract mode.

Subtract
: Pixels are subtracted from one another. Gives a dark result.
: The "opposite" to Add mode.

Multiply
: Returns a darker result than either pixel in most cases (except if one of them equals white=1). Completely white layers do not

change the background at all. Completely black layers give a black result.
The "opposite" to Screen mode.

Screen
: Both pixel values are inverted, multiplied by each other, then the result is inverted again. This returns a brighter result than both input pixels in most cases (except if one of them equals 0). Completely black layers do not change the background at all (and vice versa); completely white layers give a white result.
The "opposite" of Multiply mode.

Overlay
: A combination of Screen and Multiply mode, depending on the base color.

Divide
: The background pixel (top socket) is divided by the second one: if this one is white (= 1.0), the first one isn't changed; the darker the second one, the brighter is the result (division by 0.5 - median gray - is same as multiplication by 2.0); if the second is black (= 0.0, zero-division is impossible!), Blender doesn't modify the background pixel.

Difference
: Both pixels are subtracted from one another, and the absolute value is taken. So the result shows the distance between both parameters, black stands for equal colors, white for opposite colors (one is black, the other white). The result looks a bit strange in many cases. This mode can be used to invert parts of the base image, and to compare two images (results in black if they are equal).

Darken
: Both pixels are compared to each other, and the smaller one is taken. Completely white layers do not change the background at all, and completely black layers give a black result.

Lighten
: Both parameters are compared to each other, and the larger one is taken. Completely black layers do not change the image at all and white layers give a white result.

Dodge
: Brightens the one socket by the gradient in the other socket. Results in lighter areas of the image where the gradient is whiter. Use the Fac to control how much the gradient affects the other socket.

Burn
: Darkens one socket based on the gradient fed to the other socket. Results in darker images, since the image is *burned* onto the paper, er ... image (showing my age).

Color
: Adds a color to a pixel, tinting the overall whole with the color. Use this to increase the tint of an image.

Value
: The RGB values of both pixels are converted to HSV values. The values of both pixels are blended, and the hue and saturation of the base image is combined with the blended value and converted back to RGB.

Saturation
: The RGB values of both pixels are converted to HSV values. The saturation of both pixels are blended, and the hue and value of the base image is combined with the blended saturation and converted back to RGB.

Hue
: The RGB values of both pixels are converted to HSV values. The hue of both pixels are blended, and the value and saturation of the base image is combined with the blended hue and converted back to RGB.

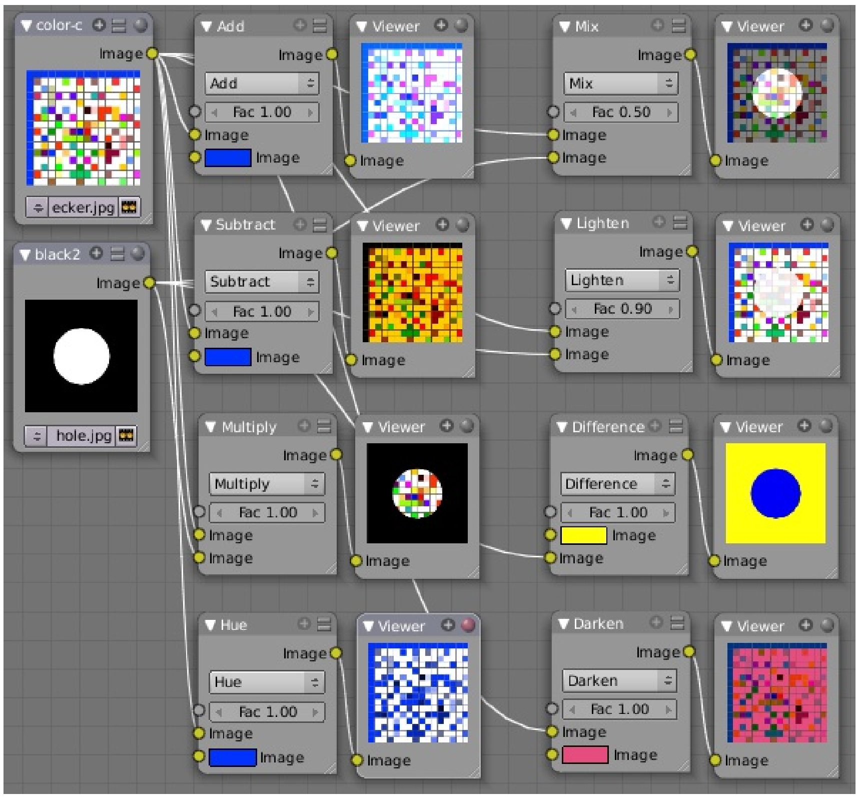
Color Channels
There are two ways to express the channels that are combined to result in a color: RGB or HSV. RGB stands for the Red/Green/Blue pixel format, and HSV stands for the Hue/Saturation/Value pixel format.

Alpha
: Click the Alpha button to make the mix node use the Alpha (transparency) values of the second (bottom) node. If enabled, the resulting image will have an Alpha channel that reflects both images' channels. Otherwise, (when not enabled, light green) the output image will mix the colors by considering what effect the Alpha channel has of the base (top input socket) image. The Alpha channel of the output image is not affected.

Fac
: The amount of mixing of the bottom socket is selected by the Factor input field (Fac:). A factor of zero does not use the bottom socket, whereas a value of 1.0 makes full use. In Mix mode, 50:50 (0.50) is an even mix between the two, but in Add mode, .50 means that only half of the second socket's influence will be applied.

Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.

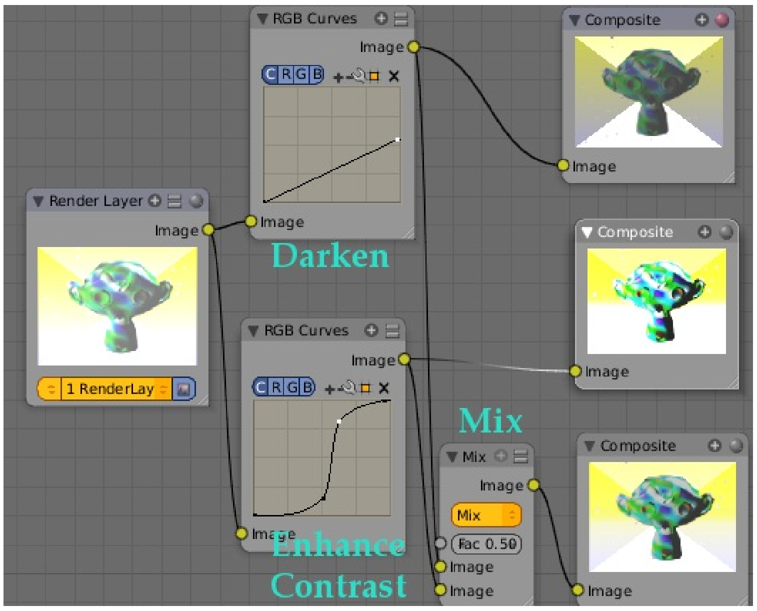


Some explanation of the mixing methods above might help you use the Mix node effectively:

- *Add* - adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.
- *Subtract*: Taking Blue away from white leaves Red and Green, which combined make Yellow (and you never thought you'd need a color wheel again, eh?). Taking Blue away from Purple leaves Red. Use this to de-saturate an image. Taking away yellow makes an image bluer and more depressing.
- *Multiply*: Black (0.00) times anything leaves black. Anything times White (1.00) is itself. Use this to mask out garbage, or to colorize a black-and-white image.
- *Hue*: Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').
- *Mix*: Combines the two images, averaging the two.
- *Lighten*: Like bleach, makes your whites whiter. Use with a mask to lighten up a little.
- *Difference*: Kinda cute in that it takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two verrry similar images to see what had been done to one to make it the other; sorta like a change log for images. You can use this to see a watermark you have placed in an image for theft detection.
- *Darken*, with the colors set here, is like looking at the world through rose-colored glasses (sorry, I just couldn't resist).

Contrast Enhancement using Mix

Here is a small map showing the effects of two other common uses for the RGB Curve: **Darken** and **Contrast Enhancement**. You can see the effect each curve has independently, and the combined effect when they are **mixed** equally.



Example node setup showing "Darken", "Enhance Contrast" and "Mix" nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast. Other paint programs usually provide a slider type of control, but Blender, ah the fantastic Blender, provides a user-definable curve to provide precise control.

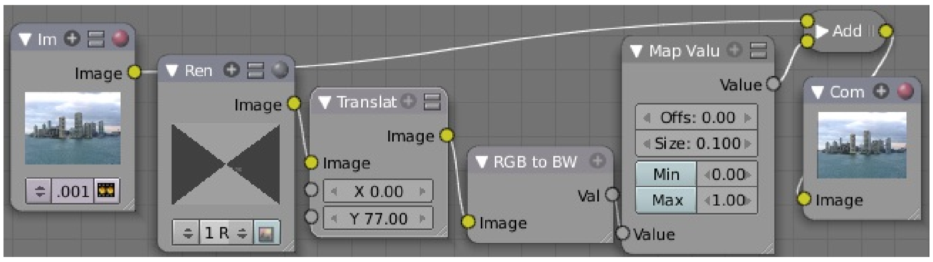
In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB 'S' curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better. And NOBODY wants a cranky monkey on their hands.

Using Mix to Watermark images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising. Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting service, or simply to track their image's proliferation throughout the web. Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding Your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it's up to you. In the example below, we are encoding the watermark in a specific location in the image using the Translate node; this helps later because we only have to look in a specific location for the mark. We then use the RGB to BW node to convert the image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, make the ones containing the mark ever-so-slightly brighter.



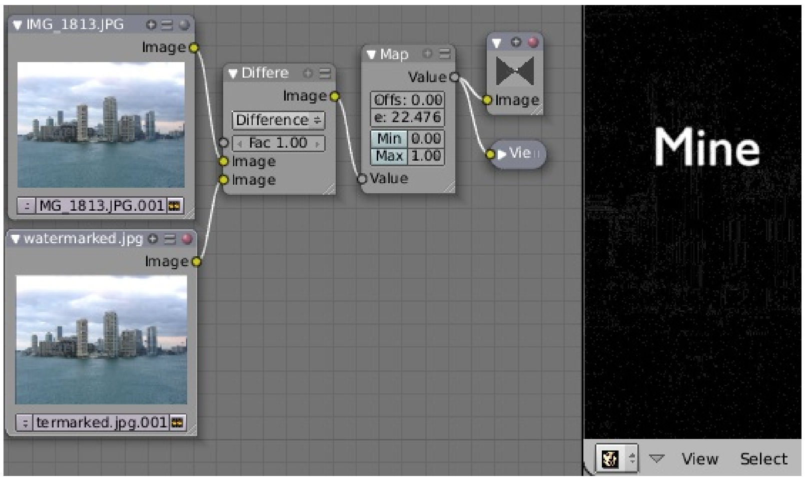
Embedding your mark in an Image using a Mark and Specific Position

Of course, if you *want* people to notice your mark, don't scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Additional uses
You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the possessed girl's forearm. yuk.

Decoding an Image for your Watermark

When you see an image that you think might be yours, use the node map below to compare it to your stock image (pre-watermarked original). In this map, the Mix node is set to Difference, and the Map Value node amplifies any difference. The result is routed to a viewer, and you can see how the original mark stands out, clear as a bell:



Checking an image for your watermark

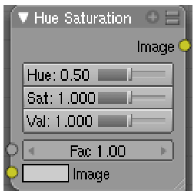
Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding map in one scene, and the decoding map in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.

Using Dodge and Burn (History Lesson)

Use the dodge and burn mix methods in combination with a mask to affect only certain areas of the image. In the old darkroom days, when, yes, I actually spent hours in a small stinky room bathed in soft red light, I used a circle cutout taped to a straw to dodge areas of the photo as the exposure was made, casting a shadow on the plate and thus limiting the light to a certain area.

To do the opposite, I would burn in an image by holding a mask over the image. The mask had a hole in it, letting light through and thus 'burning' in the image onto the paper. The same equivalent can be used here by mixing an alpha mask image with your image using a dodge mixer to lighten an area of your photo. Remember that black is zero (no) effect, and white is one (full) effect. And by the way, ya grew to like the smell of the fixer, and with a little soft music in the background and the sound of the running water, it was very relaxing. I kinda miss those dayz.

Hue Saturation Node



As an alternative to RGB editing, color can be thought of as a mix of Hues, namely a normalized value along the visible spectrum from infra-red to ultraviolet (the rainbow, remember "Roy G. Biv"). The amount of the color added depends on the saturation of that color; the higher the saturation, the more of that pigment is added. Use the saturation slider of this node to "bring out" the colors of a washed-out image.

This node takes an input image and runs the color of the image (and the light it reflects and radiates) 'up' through a factor (0.0-1.0) and applies a saturation of color effect of a hue to the image:

Hue:
: The **Hue** slider specifies how much to shift the hue of the image. Hue 0.5 (in the middle) does not shift the hue or affect the color of the image. As Hue shifts left, the colors shift as more cyan is added; a blue image goes bluer, then greener, then yellow. A red image goes violet, then purple, blue, and finally teal. Shifting right (increasing Hue from 0.5 to 1.0) introduces reds and greens. A blue image goes purple, plum, red, orange, and then yellow. A red image goes golden, olive, green, and cyan.

Sat:
: **Saturation** affect the amount of pigment in the image. A saturation of 0 actually *removes* hues from the color, resulting in a black-and-white grayscale image. A saturation of 1.0 blends in the hue, and 2.0 doubles the amount of pigment and brings out the colors.

Val:
: **Value** affects the overall amount of the color in the image. Increasing values make an image lighter; decreaing values shift an image darker.

Fac:
: **Factor** determines how much this node affects the image. A factor of 0 means that the input image is not affected by the Hue and Saturation settings. A factor of 1 means they rule, with .5 being a mix.

Hue/Saturation tips

Some things to keep in mind that might help you use this node better:

Hues are vice versa.
: A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

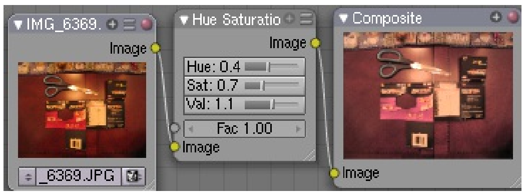
Hue and Saturation work together.
: So, a Hue of .5 keeps the blues the same shade of blue, but the saturation slider can deepen or lighten the intensity of that color.

Gray & White are neutral hues.
: A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with the Val slider. This applies for all shades of gray, from black to white; wherever the values are equal.

Changing the effect over time.
: The Hue and Saturation values are set in the node by the slider, but you can feed a Time input into the Factor to bring up (or down) the effect change over time.

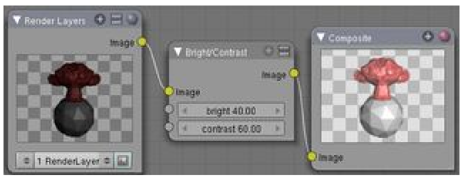
Tinge
This HSV node simply shifts hues that are already there. To colorize a gray image, or to ADD color to an image, use a mix node to add in a static color from an RGB input node with your image.

HSV Example



Here, the image taken by a cheap digital camera in poor lighting at night using a flash (can we do it any worse, eh?) is adjusted by decreasing the Hue (decreasing reds and revealing more blues and greens), decreasing Saturation (common in digital cameras, and evens out contrast) and increasing Value (making it all lighter).

Bright/Contrast

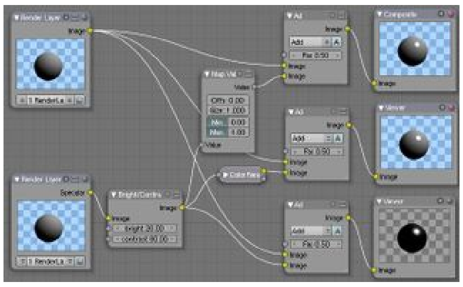


A basic example

Bright
: A multiplier-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast
: A scaling type factor by which to make brighter pixels brighter but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

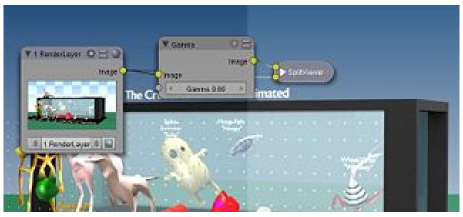
Notes



It is possible that this node will put out a value set that has values beyond normal range, i.e. values > 1 or < 0. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a ColorRamp node (with all normal defaults).

Either of these nodes will scale the values back to normal range. In the example image, we want to amp up the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has valued much less than 1 in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the ColorRamp produces the desired effect.

Gamma

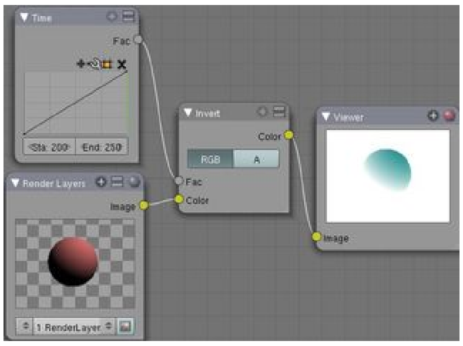


A reason for applying gamma correction to the final render is to correct lighting issues. Lighting issues that can be corrected by a gamma correction node are light attenuation with distance, light falloff at terminators, and light and shadow superpositions. Simply think about the renderer as a virtual camera. By applying a gamma correction to your render, you are just replicating what digital camera do with photos. Digital cameras gamma correct their photos, so you do the same thing. The gamma correction is, indeed, 0.45, not 2.2.

But reverse gamma correction on textures and colors have another very important consequence when you are using rendering techniques such as radiosity or GI. When doing the GI calculations, all textures and colors are taken to mean reflectance. If you do not reverse gamma correct your textures and colors, then the GI render will look way too bright because the reflected colors are all way too high and thus a lot more light is bouncing around than it should.

Gamma correction in Blender enters in a few places. The first is in this section with the nodes, both this node and the Tonemap node, and the second is in calculating Radiosity. In the noodle to the left, the split viewer shows the before and after effect of applying a gamma correction.

Invert

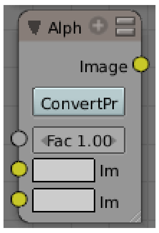


This handy node inverts the colors in the input image, producing a negative.

Options

Factor
Controls the amount of influence the node exerts on the output image
Color
The input image. In this case, a red sphere on a black transparent background
RGB
Invert the colors from white. In this example, red inverted is cyan (teal).
A
Invert the alpha (transparency) channel as well. Handy for masking.

AlphaOver Node

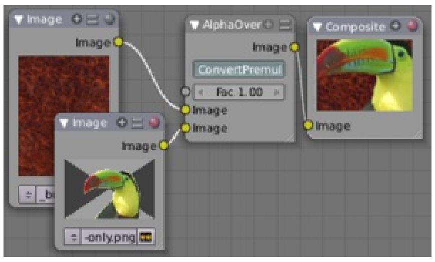


AlphaOver node

Use this node to layer images on top of one another. This node takes two images as input, combines them by a factor, and outputs the image. Connect the Background image to the top input, and the foreground image to the lower input. Where the foreground image pixels have an alpha greater than 0 (namely, have some visibility), the background image will be overlaid.

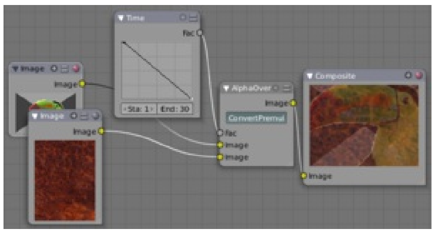
Use the Factor slider to 'merge' the two pictures. A factor less than 1.00 will make the foreground more transparent, allowing the background to bleed through.

Examples



Assembling a composite Image using AlphaOver

In this example, an image of a Toucan is superimposed over a wooden background. Use the PreMultiply button when the foreground image and background images have a combined Alpha that is greater than 1.00; otherwise you will see an unwanted halo effect. The resulting image is a composite of the two source images.



Animated See-Through/Sheer SFX using AlphaOver - Frame 11

In this example, we use the Factor control to make a sheer cloth or onion-skin effect. You can animate this effect, allowing the observer to 'see-through' walls (or any foreground object) by hooking up a Time node to feed the Factor socket as shown below. In this example, over the course of 30 frames, the Time node makes the AlphaOver node produce a picture that starts with the background wood image, and slowly bleeds through the Toucan. This example shows frame 11 just as the Toucan starts to be revealed. AlphaOver does not work on the colors of an image, and will not output any image when one of the sockets is unconnnected.

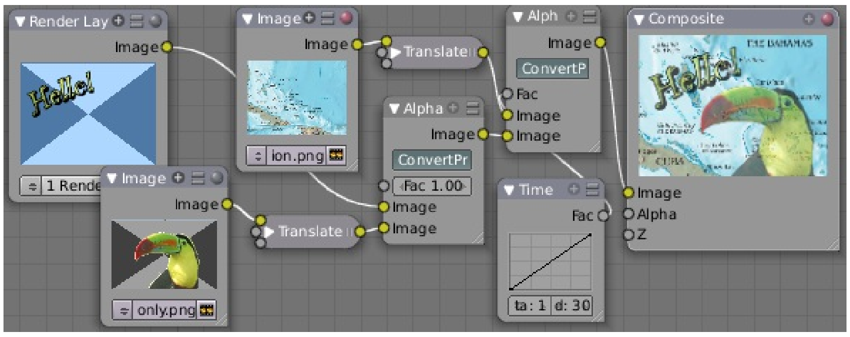
Strange Halos or Outlines

To clarify the premultiplied-alpha button: An alpha channel has a value of between 0 and 1. When you make an image transparent (to composite it over another one), you are really multiplying the RGB pixel values by the alpha values (making the image transparent (0) where the alpha is black (0), and opaque (1) where it is white (1)).

So, to composite image A over image B, you get the alpha of image A and multiply it by image A, thus making the image part of A opaque and the rest transparent. You then inverse the alphas of A and multiply image B by it, thus making image B transparent where A is opaque and vice versa. You then add the resultant images and get the final composite.

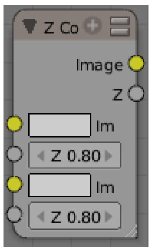
A pre-multiplied alpha is when the image (RGB) pixels are already multiplied by the alpha channel, therefore the above compositing op doesn't work too well, and you have to hit 'convert pre-mult'. This is only an issue in semi transparent area, and edges usually. The issue normally occurs in Nodes when you have combined, with alpha, two images, and then wish to combine that image with yet another image. The previously combined image was previously multiplied (pre-mult) and needs to be converted as such (hence, *Convert PreMul*).

If you don't pay attention and multiply twice, you will get a white or clear halo around your image where they meet, since your alpha value is being squared or cubed. It also depends on whether or not you have rendered your image as a pre-mult, or straight RGBA image.



Layering Images using AlphaOver Premul

Z-Combine Node

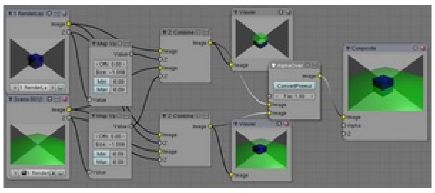


Z Combine node

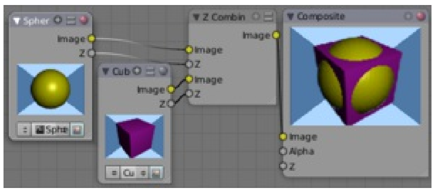
The Z-Combine node takes two images and two Z-value sets as input. It overlays the images using the provided Z values to detect which parts of one image are in front of the other. If both Z values are equal, it uses the top image. It puts out the combined image, with the combined Z-depth map, allowing you to thread multiple Z-combines together.

Z-Combine chooses whichever Z-value is less when deciding which image pixel to use. Normally, objects are in front of the camera and have a positive Z value. If one Z-value is negative, and the other positive, Z-Combine will use the image corresponding to the negative value. You can think of a negative Z value as being behind the camera. When choosing between two negative Z-values, Z-Combine will use whichever is more negative.

Alpha values carry over from the input images. Not only is the image pixel chosen, but also its alpha channel value. So, if a pixel is partially or totally transparent, the result of the Z-Combine will also be partially transparent; in which case the background image will show through the foreground (chosen) pixel. Where there are sharp edges or contrast, the alpha map will automatically be anti-aliased to smooth out any artifacts.
However, you can obtain this by making an AlphaOver of two Z-Combine, one normal, the other having inverted (reversed?) Z-values as inputs, obtained using for each of them a MapValue node with a Size field set to -1.0:


Alpha and Z-Combine node.

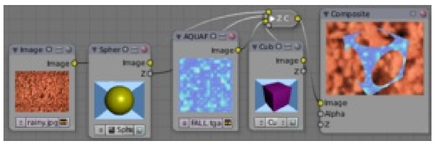
Examples


Choosing closest pixels

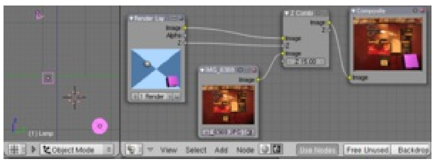
In the example to the right, render output from two scenes are mixed using the Z-Offset node, one from a sphere of size 1.30, and the other a cube of size 1.00. The sphere and square are located at the same place. The cube is tipped forward, so the corner in the center is closer to the camera than the sphere surface; so Z-Offset chooses to use the cube's pixels. But the sphere is slightly larger (a size of 1.30 versus 1.00), so it does not fit totally 'inside' the cube. At some point, as the cube's sides recede back away from the camera, the sphere's sides are closer. When this happens, Z-offset uses the sphere's pixels to form the resulting picture.

This node can be used to combine a foreground with a background matte painting. Walt Disney pioneered the use of multi-plane mattes, where three or four partial mattes were painted on glass and placed on the left and right at different Z positions; mininal camera moves to the right created the illusion of depth as Bambi moved through the forest.

Valid Input
Z Input Sockets do not accept fixed values; they must get a vector set (see Map Value node). Image Input Sockets will not accept a color, since it does not have UV coordinates.


Mix and Match Images

You can use Z-Combine to merge two images as well, using the Z-values put out by two renderlayers. Using the Z-values from the sphere and cube scenes above, but threading different images, yields the example to the right.


Z-Combine in action

In this noodle (you may click the little expand-o-matic icon in the bottom right to view it to full size), we mix a render scene with a flat image. In the side view of the scene, the purple cube is 10 units away from camera, and the gray ball is 20. The 3D cursor is about 15 units away from camera. We Z-in the image at a location of 15, thus inserting it in-between the cube and the ball. The resulting image appears to have the cube on the table.

Invisible Man Effect
If you choose a foreground image which has a higher Alpha than the background, and then mix the Z-combine with a slightly magnified background, the outline of the transparent area will distort the background, enough to make it look like you are seeing part of the background through an invisible yet Fresnel-lens object.

Color Balance

The Color Balance node can adjust the color and values of an image using two different correction formulas.

The Lift, Gammma, Gain formula uses Lift, Gamma, and Gain calculations to adjust an image.Lift increases the value of dark colors, Gamma will adjust midtones, and Gain adjusts highlights.

The Offset, Power, Slope formula uses Offset, Power, and Slope

```
 out = (i*s+o)^p,
```

where
- out = the color graded pixel code value
- i = the input pixel code value (0=black, 1=white)
- s = slope (any number 0 or greater, nominal value is 1.0)
- o = offset (any number, nominal value is 0)
- p = power (any number greater than 0, nominal value is 1.0)

Factor
: Controls the amount of influence the node exerts on the output image

Hue Correct

The Hue Correct node is able to adjust the Hue, Saturation, and Value of an image, with an input curve.

By default, the curve is a straight line, meaning there is no change. The spectrum allows you to raise or lower HSV levels for each range of pixel colors. To change a H, S, or V level, move the curve points up or down. Pixels with hue values each point in the horizontal position of the graph will be changed depending on the shape of the curve.

Tone Map

Tone mapping is a technique used in image processing and computer graphics to map one set of colors to another in order to approximate the appearance of high dynamic range images in a medium that has a more limited dynamic range.

Essentially, tone mapping addresses the problem of strong contrast reduction from the scene values (radiance) to the displayable range while preserving the image details and color appearance important to appreciate the original scene content.

The Tone Map node has two methods of calculation:

Rh Simple
- Key
  - The value the average luminance is mapped to.
- Offset
  - Normally always 1, but can be used as an extra control to alter the brightness curve
- Gamma
  - If not used, set to 1

R/D Photoreceptor
- Intensity
  - If less than zero, darkens image; otherwise, makes it brighter
- Contrast
  - Set to 0 to use estimate from input image
- Adaptation
  - If 0, global; if 1, based on pixel intensity
- Color Correction
  - If 0, same for all channels; if 1, each independent

Composite Vector Nodes

Vector Nodes

Vector nodes manipulate information about how light interacts with the scene, multiplying vector sets, and other wonderful things that normal humans barely comprehend (except math geniuses, who may not be considered 'normal'). Even if you aren't a math wiz, you'll find these nodes to be very useful.

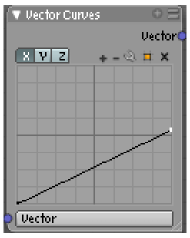
Vectors, in general, are two or three element values, for example, RGB color value, and surface normals are vectors. Vectors are also important for calculating shading.

Normal Node

The Normal node generates a normal vector and a dot product. Click and Drag on the sphere to set the direction of the normal.

This node can be used to input a new normal vector into the mix. For example, use this node as an input to a Color Mix node. Use an Image input as the other input to the Mixer. The resulting colorized output can be easily varied by moving the light source (click and dragging the sphere).

Vector Curves Node

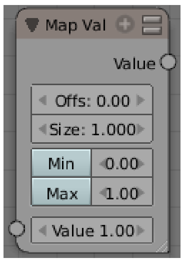


The Vector Curves node maps an input vector image's x, y, and z components to a diagonal curve. The three channels are accessed via the X, Y, and Z buttons at the top of the node. Add points to the curve by clicking on it.

Note that dragging a point across another will switch the order of the two points (e.g. if point A is dragged across point B, then point B will become point A and point A will become point B).

Use this curve to slow things down or speed them up from the original scene.

Map Value Node



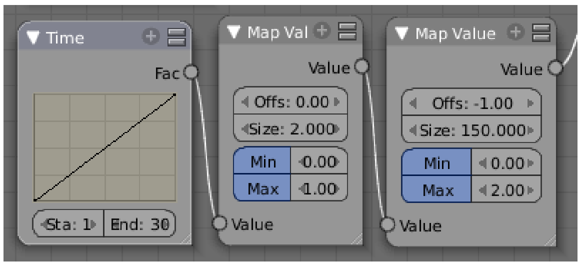
Map Value node

Map Value node is used to scale, offset and clamp values (value refers to each vector in the set). The formula for how this node works is:

- Offs will add a number to the input value
- Size will scale (multiply) that value by a number
- By clicking Min/Max you can set the minimum and maximum numbers to clamp (cut off) the value too. Min and Max must be individually enabled by LMB clicking on the label for them to clamp. ⇧ Shift LMB on the value to change it.
  - If Min is enabled and the value is less than Min, set the ouput value to Min
  - If Max is enabled and the input value is greater than Max, set the ouput value to Max

This is particularly useful in achieving a depth-of-field effect, where you can use the Map Value node to map a Z value (which can be 20 or 30 or even 500 depending on the scene) to to range between 0-1, suitable for connecting to a Blur node.

Using Map Value to Multiply values



You can also use the map value node to multiply values to achieve an output number that you desire. In the mini-map to the right, the Time node ouputs a value between 0.00 and 1.00 evenly scaled over 30 frames. The *first* Map Value node multiplies the input by 2, resulting in an output value that scales from 0.00 to 2.00 over 30 frames. The *second* Map Value node subtracts 1 from the input, giving working values between -1.00 and 1.00, and multiplies that by 150, resulting in an output value between -150 and 150 over a 30-frame sequence.

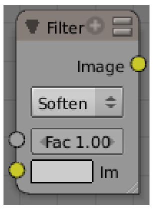
Normalize

Normalizing a vector scales its magnitude, or length, to a value of 1, but keeps its direction intact.

Composite Filter Nodes

Filters process the pixels of an image to highlight additional details or perform some sort of post-processing effect on the image.

Filter Node



Filter node

The Filter node implements various common image enhancement filters. The supported filters are, if not obvious, named after the mathematical genius who came up with them:

Soften
: Slightly blurs the image.

Sharpen
: Increases the contrast, especially at edges

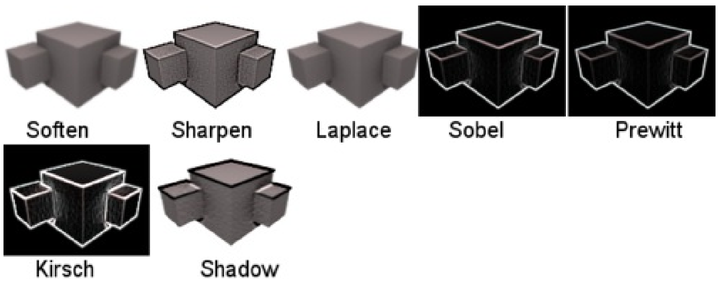
Laplace
: Softens around edges

Sobel
: Creates a negative image that highlights edges

Prewitt
: Tries to do Sobel one better.

Kirsch
: Improves on the work done by those other two flunkies, giving a better blending as you approach an edge.

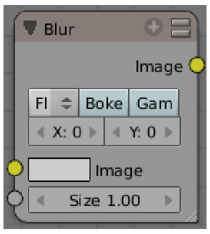
Shadow
: Performs a relief emboss/bumpmap effect, darkening outside edges.



The Filter node has seven modes, shown here.

The Soften, Laplace, Sobel, Prewitt and Kirsch all perform edge-detection (in slightly different ways) based on vector calculus and set theory equations that would fill six blackboards with gobbledy gook. Recommended reading for insomniacs.

Blur Node



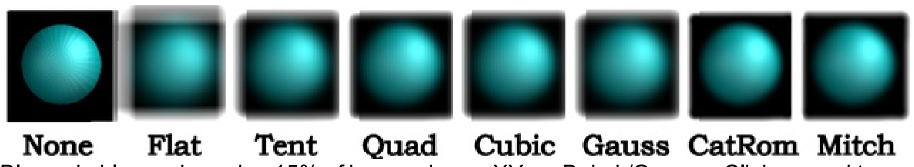
Blur node

The Blur node blurs an image, using one of seven blur modes (set using the upper-left popup button), and a radius defined by the X and Y number buttons. By default these are set to zero, so to enable the node you must set one or both to a value greater then 0. You can optionally connect a value image to the Size input node, to control the blur radius with a mask. The values must be mapped between 0-1 for best effect, as they will be multiplied with the X and Y number button values.

Options

The X and Y values are the number of pixels over which to spread the blur effect.

The Bokeh button (only visible as Bok or Bo on some screen setups) will force the blur node to use a circular blur filter. This gives higher quality results, but is slower then using a normal filter. The Gam button (for "gamma") makes the Blur node gamma-correct the image before blurring it.



Blur node blur modes using 15% of image size as XY, no Bokeh/Gamma. Click expand to see details

The difference between them is how they handle sharp edges and smooth gradients and preserve the highs and the lows. In particular (and you may have to closely examine the full-resolution picture to see this):

Flat just blurs everything uniformly
Tent preserves the high and the lows better making a linear falloff
Quadratic and CatRom keep sharp-contrast edges crisp
Cubic and Mitch preserve the highs but give almost a out-of-focus blur while smoothing sharp edges

Directional Blur Node

Blurs an image in a specified direction and magnitude. Can be used to fake motion blur.

Options

Iterations
: Controls how may times the image is duplicated to create the blur effect. Higher values give smoother results.

Wrap
: Wraps the image on the X and Y axis to fill in areas that become transparent from the blur effect.

Center
: Sets the position where the blur center is. This makes a difference if the angle, spin, and/or zoom are used.

Distance
: How large the blur effect is.

Angle
: Image is blurred at this angle from the center

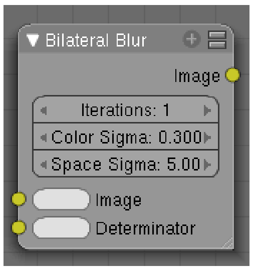
Spin
: Rotates the image each iteration to create a spin effect, from the center point.

Zoom
: Scales the image each iteration, creating the effect of a zoom.

Example

An example blend file, in fact the one used to create the image above, is available here. The .blend file takes one image from the RenderLayer "Blurs" and blurs it while offsetting it (Translate) and then combining it (AlphaOver) to build up the progressive sequence of blurs. Play with the Value and Multiply nodes to change the amount of blurring that each algorithm does.

Bilateral Blur Node



Blur node

The bilateral blur node performs a high quality adaptive blur on the source image. It can be used for various purposes like:

smoothing results from blenders raytraced ambient occlusion
smoothing results from various unbiased renderers,
to fake some performance-heavy processes, like blurry refractions/reflections, soft shadows,
to make non-photorealistic compositing effects.

Inputs

Bilateral blur has 2 inputs:

Image, for the image to be blurred.
Determinator, which is non-obligatory, and is used only if connected.

if only 1st input is connected, the node blurs the image depending on the edges present in the source image. If the Determinator is connected, it serves as the source for defining edges/borders for the blur in the image. This has great advantage in case the source image is too noisy, but normals in combination with zbuffer can still define exact borders/edges of objects.

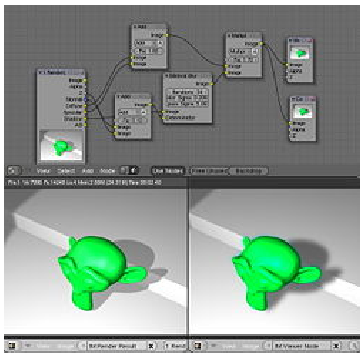
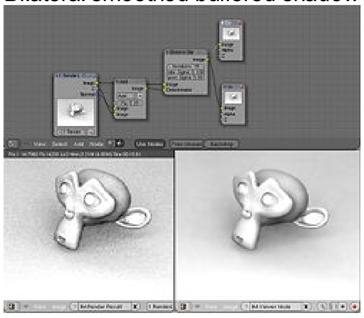
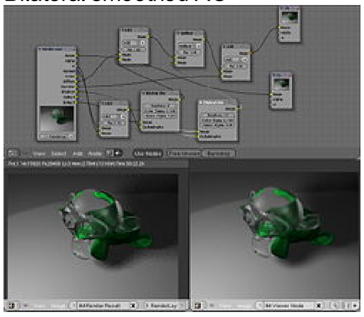
Options

Iterations
: Defines how many times the filter should perform the operation on the image. It practically defines the radius of blur.

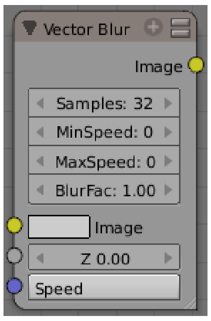
Color Sigma
: Defines the threshold for which color differences in the image should be taken as edges.

Space sigma
: A fine-tuning variable for blur radius.

Examples


Bilateral smoothed buffered shadow

Bilateral smoothed AO

Bilateral faked blurry refraction+smoothed reytraced soft shadow

Vector (Motion) Blur Node



Vector Blur node

Motion blur is the effect of objects moving so fast they blur. Because CG animations work by rendering individual frames, they have no real knowledge of what was where in the last frame, and where it is now.

In Blender, there are two ways to produce motion blur. The first method (which produces the most correct results) works by rendering a single frame up to 16 times with slight time offsets, then accumlating these images together; this is called Motion Blur and is activated on the Render panel. The second (and much faster) method is the Compositor node Vector Blur.

To use, connect the appropriate passes from a Render Result node.

Note
Make sure to enable the Speed (called Vec) pass in the Render Layers panel for the render layer you wish to perform motion blur on.

Maximum Speed: Because of the way vector blur works, it can produce streaks, lines and other artifacts. These mostly come from pixels moving too fast; to combat these problems, the filter has minimum and maximum speed settings, which can be used to limit which pixels get blurred (e.g. if a pixel is moving really, really fast but you have maximum speed set to a moderate amount, it won't get blurred).

Minimum Speed: Especially when the camera itself moves, the mask created by the vectorblur node can become the entire image. A very simple solution is to introduce a small threshold for moving pixels, which can efficiently separate the hardly-moving pixels from the moving ones, and thus create nice looking masks. You can find this new option as 'min speed'. This minimum speed is in pixel units. A value of just 3 will already clearly separate the background from foreground.

Hint
You can make vector blur results a little smoother by passing the Speed pass through a blur node (but note that this can make strange results, so it's only really appropriate for still images with lots of motion blur).

Examples

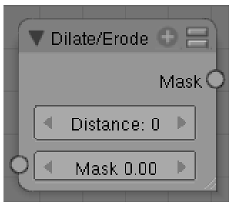
An in-depth look at how to use the Vector Blur node can be found here.

As far as we know, this node represents a new approach to calculating motion blur. Use vector blur in compositing with confidence instead of motion blur. In face, when compositing images, it is necessary to use vector blur since there isn't "real" motion. In this example blend file, you will find a rigged hand reaching down to pick up a ball. Based on how the hand is moving (those vectors), the image is blurred in that direction. The fingers closest to the camera (the least Z value) are blurred more, and those farther away (the forearm) is blurred the least.

Known Bugs

Does not work when reading from a multilayer OpenEXR sequence set	Blender 2.44+

Dilate/Erode Node

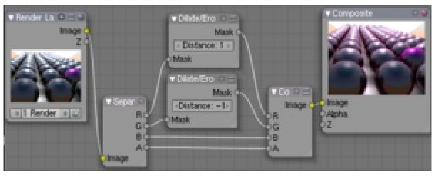


Dilate/Erode
Mask
Distance: 0
Mask 0.00

Dilate/Erode node

This node blurs individual color channels. The color channel (or a black and white image) is connected to the Mask input socket, and the Distance is set manually (by clicking on the arrows or the value) or automatically from a value node or a time-and-map-value noodle. A positive value of Distance expands the influence of a pixel on its surrounding pixels, thus blurring that color outward. A negative value erodes its influence, thus increases the constrast of that pixel relative to its surrounding pixels, thus sharpening it relative to surrounding pixels of the same color.

Example



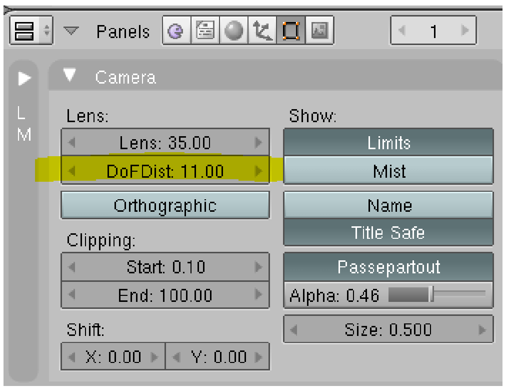
Magenta tinge

In the above example image, we wanted to take the rather boring array of ball bearings and spruce it up; make it hot, baby. So, we dilated the red and eroded the green, leaving the blue alone. If we had dilated both red and green...(hint: red and green make yellow). The amount of influence is increased by increasing the Distance values. Blend file available here.

Defocus

This single node can be used to emulate depth of field using a postprocessing method. It can also be used to blur the image in other ways, not necessarily based on 'depth' by connecting something other than a Zbuffer. In essence, this node blurs areas of an image based on the input zbuffer map/mask.

Camera Settings



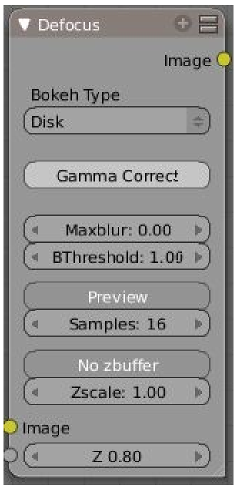
DofDist setting for the camera.

The Defocus node uses the actual camera data in your scene if supplied by a RenderLayer node.

To set the point of focus, the camera now has a Distance parameter, which is shorthand for Depth of Field Distance. Use this camera parameter to set the focal plane of the camera (objects Depth of Field Distance away from the camera are in focus). Set Distance in the main Camera edit panel; the button is right below the Depth of Field.

To make the focal point visible, enable the camera Limits option, the focal point is then visible as a yellow cross along the view direction of the camera.

Node Inputs



Defocus node

The node requires two inputs, an image and a zbuffer, the latter does not need to be an actual zbuffer, but can also be another (grayscale) image used as mask, or a single value input, for instance from a time node, to vary the effect over time.

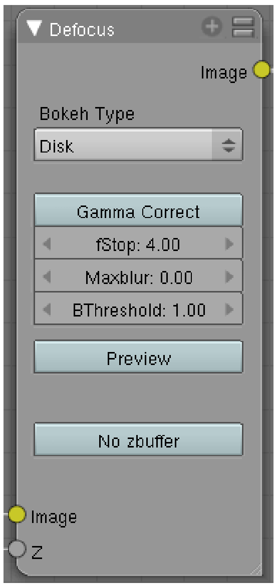
Node Setting

The settings for this node are:

Bokeh Type menu
: Here you set the number of iris blades of the virtual camera's diaphragm. It can be set to emulate a perfect circle (Disk) or it can be set to have 3 (Triangle), 4 (Square), 5 (Pentagon), 6 (Hexagon), 7 (Heptagon) or 8 blades (Octagon). The reason it does not go any higher than 8 is that from that point on the result tends to be indistinguishable from a Disk shape anyway.

Rotate
: This button is not visible if the Bokeh Type is set to Disk. It can be used to add an additional rotation offset to the Bokeh shape. The value is the angle in degrees.

Gamma Correct
: Exactly the same as the Gamma option in Blender's general Blur node (see Blur Node). It can be useful to further brighten out of focus parts in the image, accentuating the Bokeh effect.



Defocus node using Z-Buffer

fStop
: This is the most important parameter to control the amount of focal blur: it simulates the aperture *f* of a real lens(' iris) - without modifying the luminosity of the picture, however! As in a real camera, the *smaller* this number is, the more-open the lens iris is, and the *shallower* the depth-of-field will be. The default value 128 is assumed to be infinity: everything is in perfect focus. Half the value will double the amount of blur. This button is not available if No zbuffer is enabled.

Maxblur
: Use this to limit the amount of blur of the most out of focus parts of the image. The value is the maximum blur radius allowed. This can be useful since the actual blur process can sometimes be very slow. (The more blur, the slower it gets.) So, setting this value can help bring down processing times, like for instance when the world background is visible, which in general tends to be the point of maximum blur (not always true, objects very close to the lens might be blurred even more). The default value of 0 means there is no limit to the maximum blur amount.

BThreshold
: The defocus node is not perfect: some artifacts may occur. One such example is in-focus objects against a blurred background, which have a tendency to bleed into the edges of the sharp object. The worst-case scenario is an object in-focus against the very distant world background: the differences in distance are very large and the result can look quite bad. The node tries to prevent this from occurring by testing that the blur difference between pixels is not too large, the value set here controls how large that blur difference may be to consider it 'safe.' This is all probably quite confusing, and fortunately, in general, there is no need to change the default setting of 1. Only try changing it if you experience problems around any in-focus object.

Preview
: As already mentioned, processing can take a long time. So to help make editing parameters somewhat 'interactive', there is a preview mode which you can enable with this button. Preview mode will render the result using a limited amount of (quasi)random samples, which is a *lot* faster than the 'perfect' mode used otherwise. The sampling mode also tends to produce grainy, noisy pictures (though the more samples you use, the less noisy the result). This option is on by default. Play around with the other parameters until you are happy with the results, and only then disable the preview mode for the final render.

Samples
: Only visible when Preview is set. Sets the amount of samples to use to sample the image. The higher, the smoother the image, but also the longer the processing time. For preview, the default of 16 samples should be sufficient and is also the fastest.

No zbuffer
: Sometimes you might want to have more control to blur the image. For instance, you may want to only blur one object while leaving everything else alone (or the other way around), or you want to blur the whole image uniformly all at once. The node therefore allows you to use something other than an actual zbuffer as the Z input. For instance, you could connect an image node and use a grayscale image where the color designates how much to blur the image at that point, where white is maximum blur and black is no blur. Or, you could use a Time node to uniformly blur the image, where the time value controls the maximum blur for that frame. It may also be used to obtain a possibly slightly-better DoF blur, by using a fake depth shaded image instead of a zbuffer. (A typical method to create the fake depth shaded image is by using a linear blend texture for all objects in the scene or by using the 'fog/mist' fake depth shading method.) This also has the advantage that the fake depth image can have anti-aliasing, which is not possible with a real zbuffer. "No zbuffer" will be enabled automatically whenever you connect a node that is not image based (e.g. time node/value node/etc).

Zscale
: Only visible when No zbuffer enabled. When No zbuffer is used, the input is used directly to control the blur radius. And since usually the value of a texture is only in the numeric range 0.0 to 1.0, its range is too narrow to control the blur properly. This parameter can be used to expand the range of the input (or for that matter, narrow it as well, by setting it to a value less than one). So for No zbuffer, this parameter therefore then becomes the main blur control (similar to fStop when you *do* use a zbuffer).

Examples

In this blend file example, the ball array image is blurred as if it was taken by a camera with a f-stop of 2.8 resulting in a farily narrow depth of field centered on 7.5 blender units from the camera. As the balls receed into the distance, they get blurrier.

Hints

Preview
: In general, use preview mode, change parameters to your liking, only then disable preview mode for the final render. This node is compute intensive, so watch your console window, and it will give you status as it computes each render scan line.

Edge Artifacts
: For minimum artifacts, try to setup your scene such that differences in distances between two objects that may visibly overlap at some point are not too large.

"Focus Pull"
: Keep in mind that this is not 'real' DoF, only a post-processing simulation. Some things cannot be done which would be no problem for real DoF at all. A typical example is a scene with some object very close to the camera, and the camera focusing on some point far behind it. In the real world, using shallow depth of field, it is not impossible for nearby objects to become completely invisible, in effect allowing the camera to see 'behind' it. Hollywood cinematographers use this visual characteristic to good effect to achieve the popular "focus pull" effect, where the focus shifts from a nearby to a distant object, such that the "other" object all but disappears. Well, this is simply not possible to do with the current post-processing method in a single pass. If you really want to achieve this effect, quite satisfactorily, here's how:

- Split up your scene into "nearby" and "far" objects, and render them in two passes.
- Now, combine the two the two results, each with their own "defocus" nodes driven by the same Time node, but with one of them inverted. (e.g. using a "Map Value" node with a Size of -1.) As the defocus of one increases, the defocus on the other decreases at the same rate, creating a smooth transition.

Aliasing at Low f-Stop Values
: At very low values, less than 5, the node will start to remove any oversampling and bring the objects at DoFDist very sharply into focus. If the object is against a constrasting background, this may lead to visible stairstepping (aliasing) which OSA is designed to avoid. If you run into this problem:

- Do your own OSA by rendering at twice the intended size and then scaling down, so that adjacent pixels are blurred togther
- Use the blur node with a setting of 2 for x and y
- Set DoFDist off by a little, so that the object in focus is blurred by the tiniest bit.
- Use a higher f-Stop, which will start the blur, and then use the Z socket to a Map Value to a Blur node to enhance the blur effect.
- Rearrange the objects in your scene to use a lower-contrast background

No ZBuffer
: A final word of warning, since there is no way to detect if an actual zbuffer is connected to the node, be VERY careful with the "No ZBuffer" switch. If the Zscale value happens to be large, and you forget to set it back to some low value, the values may suddenly be interpreted as huge blur-radius values that will cause processing times to explode.

Composite Convertor Nodes

As the name implies, these nodes convert the colors or other properties of various data (e.g. images) in some way. They also split out or re-combine the different color channels that make up an image, allowing you to work on each channel independently. Various color channel arrangements are supported, including traditional RGB and HSV formats, and the newest High Definition Media Interface (HDMI) formats.

ColorRamp Node

The ColorRamp Node is used for mapping values to colors with the use of a gradient. It works exactly the same way as a colorband for textures and materials, using the Factor value as a slider or index to the color ramp shown, and outputting a color value and an alpha value from the output sockets.

By default, the ColorRamp is added to the node map with two colors at opposite ends of the spectrum. A completely black black is on the left (Black as shown in the swatch with an Alpha value of 1.00) and a whitewash white is on the right. To select a color, LMB click on the thin vertical line/band within the colorband. The example picture shows the black color selected, as it is highlighted white. The settings for the color are shown above the colorband as (left to right): color swatch, Alpha setting, and interpolation type.

To change the hue of the selected color in the colorband, LMB click on the swatch, and use the popup color picker control to select a new color. Press ↵ Enter to set that color.

To add colors, hold Ctrl down and Ctrl LMB click inside the gradient. Edit colors by clicking on the rectangular color swatch, which pops up a color-editing dialog. Drag the gray slider to edit Alpha values. Note that you can use textures for masks (or to simulate the old "Emit" functionality) by connecting the alpha output to the factor input of an RGB mixer.

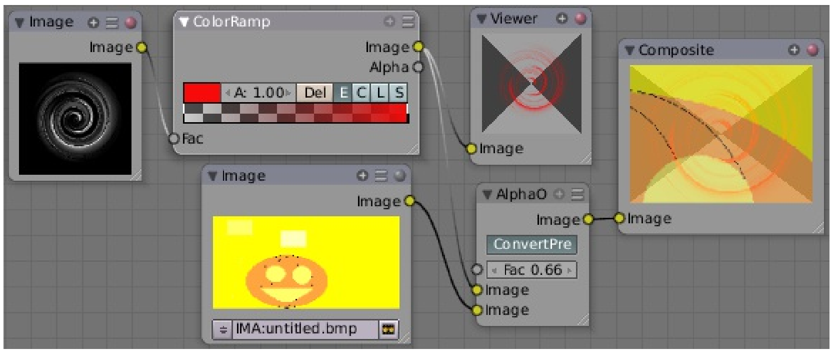
To delete a color from the colorband, select it and press the Delete button.

When using multiple colors, you can control how they transition from one to another through an interpolation mixer. Use the interpolation buttons to control how the colors should band together: Ease, Cardinal, Linear, or Spline.

Use the A: button to define the Alpha value of the selected color for each color in the range.

Using ColorRamp to create an Alpha Mask

A powerful but often overlooked feature of the ColorRamp is to create an Alpha Mask, or a mask that is overlaid on top of another image, and, like a mask, allows some of the background to show through. The example map below shows how to use the Color Ramp node to do this:



Using the ColorRamp node to create an alpha mask

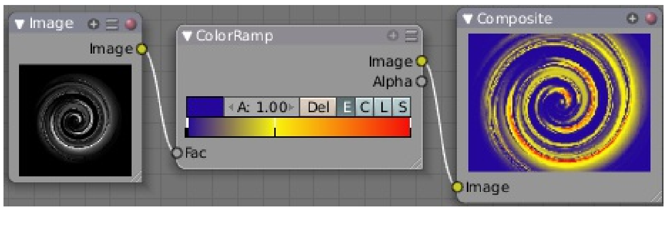
In the map above, a black and white swirl image, which is lacking an alpha channel, is fed into the ColorRamp node as a Factor. (Technically, we should have converted the image to a value using the RGB-to-BW node, buy hey, this works just as well since we are using a BW image as input.)

We have set the ColorRamp node to a purely transparent color on the left end of the spectrum, and a fully Red color on the right. As seen in the viewer, the ColorRamp node puts out a mask that is fully transparent where the image is black. Black is zero, so ColorRamp uses the 'color' at the left end of the spectrum, which we have set to transparent. The ColorRamp image is fully red and opaque where the image is white (1.00).

We verify that the output image mask is indeed transparent by overlaying it on top of a pumpkin image. For fun, we made that AlphaOver output image 0.66 transparent so that we can, in the future, overlay the image on a flashing white background to simulate a scary scene with lighting flashes.

Using ColorRamp to Colorize an Image

The real power of ColorRamp is that multiple colors can be added to the color spectrum. This example compositing map takes a boring BW image and makes it a flaming swirl!

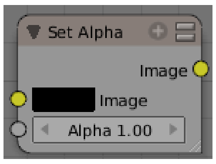


In this example, we have mapped the shades of gray in the input image to three colors, blue, yellow, and red, all fully opaque (Alpha of 1.00). Where the image is black, ColorRamp substitutes blue, the currently selected color. Where it is some shade of gray, ColorRamp chooses a corresponding color from the spectrum (bluish, yellow, to reddish). Where the image is fully white, ColorRamp chooses red.

RGB to BW Node

This node converts an RGB input and outputs a greyscale image.

Set Alpha Node



Set Alpha node

This node adds an alpha channel to a picture. Some image formats, such as JPEG, do not support an alpha channel. In order to overlay a JPEG image on top of a background, you must add an alpha channel to it using this node.

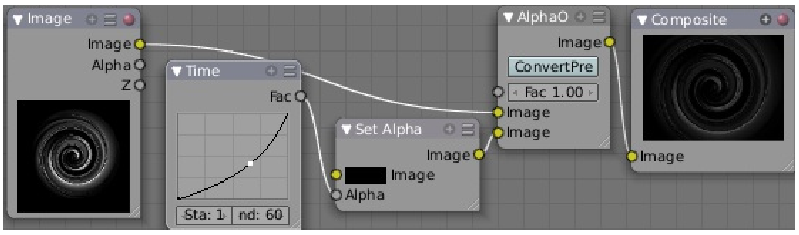
The Image input socket is optional. If an input image is not supplied, the base color shown in the swatch will be used. To change the color, LMB click the swatch and use the color-picker control to choose or specify a color you want.

The amount of Alpha (1.00 being totally opaque and 0.00 being totally transparent) can be set for the whole picture using the input field. Additionally, the Alpha factor can be set by feeding its socket.

Note that this is not, and is not intended to be, a general-purpose solution to the problem of compositing an image that doesn't contain Alpha information. You might wish to use "Chroma Keying" or "Difference Keying" (as discussed elsewhere) if you can. This node is most often used (with a suitable input being provided by means of the socket) in those troublesome cases when you *can't,* for some reason, use those techniques directly.

Using SetAlpha to Fade to Black

To transition the audience from one scene or shot to another, a common technique is to "fade to black". As its name implies, the scene fades to a black screen. You can also "fade to white' or whatever color you wish, but black is a good neutral color that is easy on the eyes and intellectually "resets" the viewer's mind. The node map below shows how to do this using the Set Alpha node.



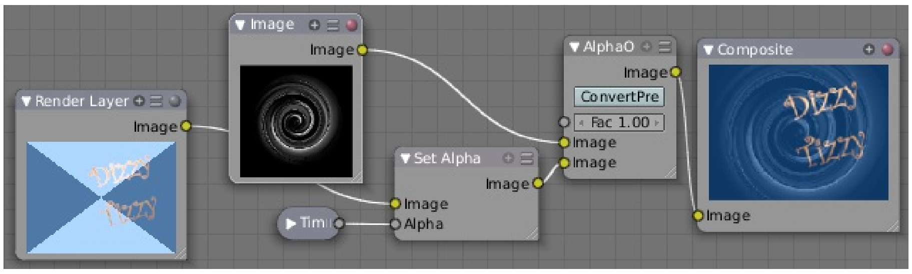
Fade To Black

In the example above, the alpha channel of the swirl image is ignored. Instead, a time node introduces a factor from 0.00 to 1.00 over 60 frames, or about 2 seconds, to the Set Alpha node. Note that the time curve is exponentially-shaped, so that the overall blackness will fade in slowly and then accelerate toward the end. The Set Alpha node does not need an input image; instead the flat (shadeless) black color is used. The Set Alpha Node uses the input factor and color to create a black image that has an alpha set which goes from 0.00 to 1.00 over 60 frames, or completely transparent to completely opaque. Think of alpha as a multiplier for how vivid you can see that pixel. These two images are combined by our trusty AlphaOver node completely (a Factor of 1.00) to produce the composite image. The SetAlpha node will thus, depending on the frame being rendered, produce a black image that has some degree of transparency. Set up and Animate, and you have an image sequence that fades to black over a 2-second period.

No Scene information used
This example node map does not use the RenderLayer. To produce this 2 second animation, no blender scene information was used. This is an example of using Blender's powerful compositing abilities separate from its modeling and animation capabilities. (A Render Layer could be substituted for the Image layer, and the "fade-network" effect will still produce the same effect)

Using SetAlpha to Fade In a Title

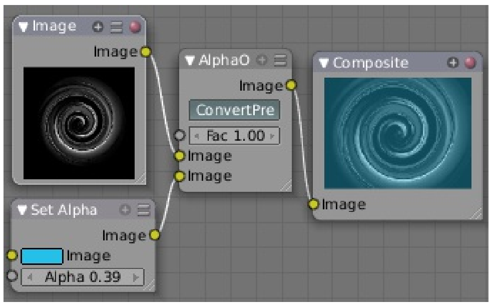
To introduce your animation, you will want to present the title of your animation over a background. You can have the title fly in, or fade it in. To fade it in, use the SetAlpha node with the Time node as shown below.



Using Set Alpha to Fade in a Title

In the above example, a Time curve provides the Alpha value to the input socket. The current RenderLayer, which has the title in view, provides the image. As before, the trusty AlphaOver node mixes (using the alpha values) the background swirl and the alphaed title to produce the composite image. Notice the ConvertPre-Multiply button is NOT enabled; this produces a composite where the title lets the background image show through where even the background image is transparent, allowing you to layer images on top of one another.

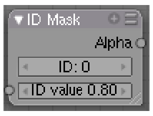
Using SetAlpha to Colorize a BW Image



Using Set Alpha to Colorize an Image
In the example above, notice how the blue tinge of the render input colors the swirl. You can use the Set Alpha node's color swatch with this kind of node map to add a consistent color to a BW image.

In the example map to the right, use the Alpha value of the SetAlpha node to give a desired degree of colorization. Thread the input image and the Set Alpha node into an AlphaOver node to colorize any black and white image in this manner. Note the ConvertPre-Multiply button is enabled, which tells the AlphaOver node not to multiply the alpha values of the two images together.

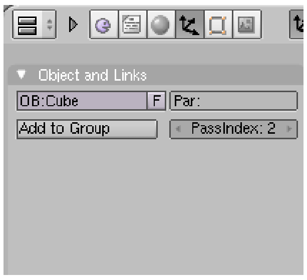
ID Mask Node



ID Mask node

This node will use the Object Index pass (see RenderLayers) to produce an anti-aliased alpha mask for the object index specified. The mask is opaque where the object is, and transparent where the object isn't. If the object is partially transparent, the alpha mask matches the object's transparency. This post-process function fills in the jaggies with interpolated values.

Object Index
Object indices are only output from a RenderLayers node or stored in a multilayer OpenEXR format image.

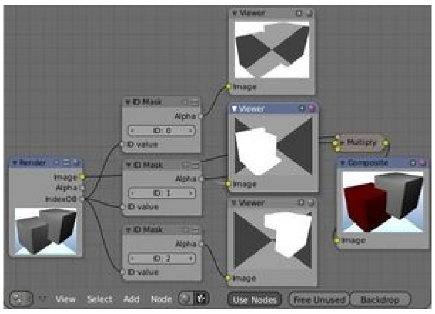


Setting an Object Index

You can specify, for any of the objects in your scene, an Object Index as shown the right (the currently select object has an index of 2). When rendered, if Object Index passes are enabled, its index will be 2, and setting the ID Mask node to 2 will show where that object is in the scene.

This node is extremely well suited to removing the aliases shown as output from the Defocus node or DOF noodles caused by some objects being close to camera against objects far away.

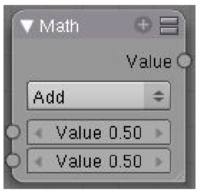
Example


Example

In this example, the left rear red cube is assigned PassIndex 1, and the right cube PassIndex 2. Where the two cubes intersect, there is going to be noticeable pixelation (jaggies) because they come together at a sharp angle and are different colors. Using the mask from object 1, which is smoothed (anti-aliased) at the edges, we use a Mix node set on Multiply to multiply the smoothed edges against the image, thus removing those nasty (Mick) Jaggies. Thus, being smoothed out, the Rolling Stones gather no moss. (I really hope you get that obscure reference :)

Note that the mask returns white where the object is fully visible to the camera(not behind anything else) and black for the part of the object that is partially or totally obscured by a fully or partially opaque object in front of it. If something else is in front of it, even if that thing is partially transparent and you can see the object in a render, the mask will not reflect that partially obscured part.

Math Node



Math node

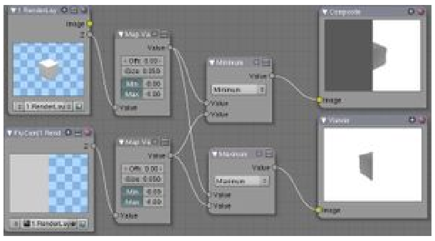
This node performs the selected math operation on an image or buffer. All common math functions are supported. If only an image is fed to one Value socket, the math function will apply the other Value consistently to every pixel in producing the output Value. Select the math function by clicking the up-down selector where the "Add" selection is shown.

The trig functions of Sine, Cosine, Tangent use only the top socket and accept values in radians between 0 and 2*pi for one complete cycle.

Known bug: the Top socket must get the image if the bottom socket is left as a value. Blender 2.44+

Examples

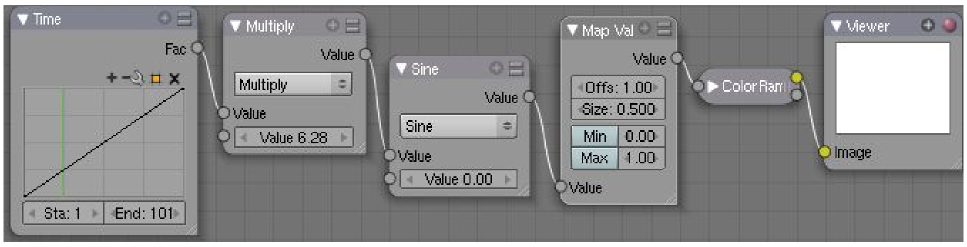
Manual Z-Mask


Example

This example has one scene input by the top RenderLayer node, which has a cube that is about 10 BU from the camera. The bottom RenderLayer node inputs a scene (FlyCam) with a plane that covers the left half of the view and is 7 BU from the camera. Both are fed through their respective Map Value nodes to divide the Z buffer by 20 (multiply by .05, as shown in the Size field) and clamped to be a Min/Max of 0.0/1.0 respectively.

For the Minimum function, the node selects those Z values where the corresponding pixel is closer to the camera; so it chooses the Z values for the plane and part of the cube. The background has an infinite Z value, so it is clamped to 1.0 (shown as white). In the maximum example, the Z values of the cube are greater than the plane, so they are chosen for the left side, but the plane (FlyCam) Renderlayer's Z are infinite (mapped to 1.0) for the right side, so they are chosen.

Using Sine Function to Pulsate

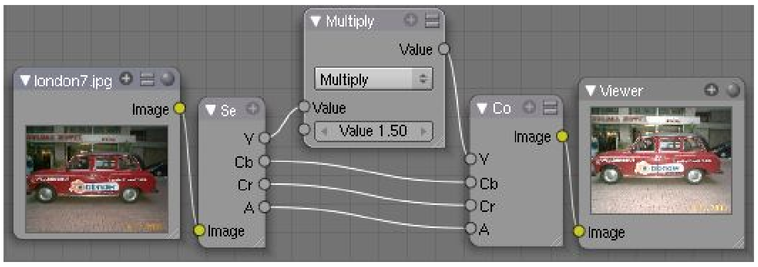


This example has a Time node putting out a linear sequence from 0 to 1 over the course of 101 frames. The green vertical line in the curve widget shows that frame 25 is being put out, or a value of .25. That value is multiplied by 2*pi and converted to 1.0 by the Sine function, since we all know that Sine(2*pi/4)=Sine(pi/2)=+1.0.

Since the Sine function can put out values between -1.0 and 1.0, the Map Value node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by one half (thus scaling the output between 0 and 1). The default ColorRamp converts those values to a grayscale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, Sine(pi/2)=1.0. Like having your own visual color calculator! Animating this noodle provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move an scene in/out of focus. Alter a color channel value to make a color "pulse".

Brightening/Scaling a Channel



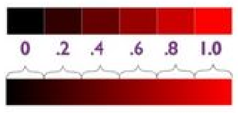
This example has a Multiply node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a Map Value node with Min() and Max() enabled to clamp the output to valid values. With this approach you could use a logarithmic function to make a high-dynamic range image. For this particular example, there is also a Brighten/Contrast node that might give simpler control over brightness.

Quantize/Restrict Color Selection

In this example, we want to restrict the color output to only 256 possible values. Possible use of this is to see what the image will look like on an 8-bit cell phone display. To do this, we want to restrict the R, G and B values of any pixel to be one of a certain value, such that when they are combined, will not result in more than 256 possible values. The number of possible values of an output is the number of channel values multiplied by each other, or Q = R * G * B.

Since there are 3 channels and 256 values, we have some flexibility how to quantize each channel, since there are a lot of combinations of R*G*B that would equal 256. For example, if {R,G,B} = {4,4,16}, then 4 * 4 * 16 = 256. Also, {6,6,7} would give 252 possible values. The difference in appearance between {4,4,16} and {6,6,7} is that the first set (4,4,16} would have fewer shades of red and green, but lots of shades of blue. The set {6,6,7} would have a more even distribution of colors. To get better image quality with fewer color values, give more possible values to the predominant colors in the image.

Theory

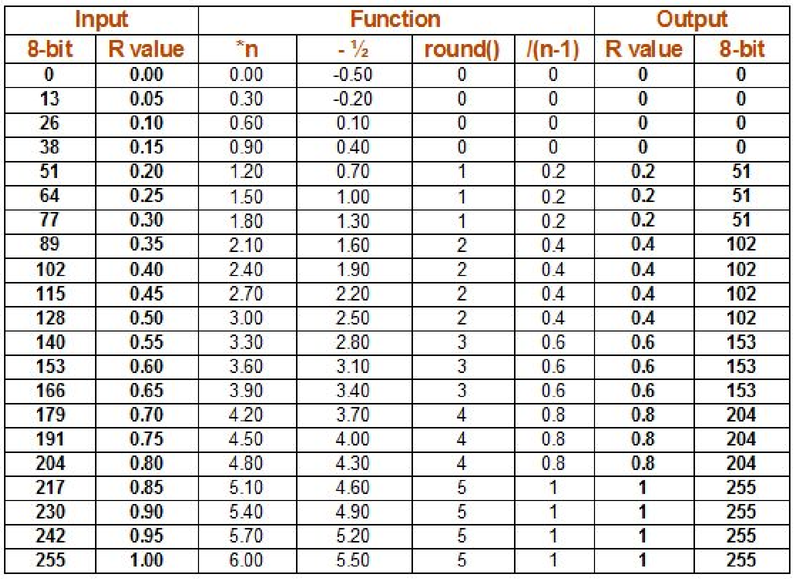


Two Approaches to Quantizing
to 6 values

To accomplish this quantization of an image to 256 possible values, lets use the set {6,6,7}. To split up a continuous range of values between 0 and 1 (the full Red spectrum) into 6 values, we need to construct an algorithm or function that takes any input value but only puts out 6 possible values, as illustrated by the image to the right. We want to include 0 as true black, with five other colors in between. The approach shown produces {0,.2,.4,.6,.8,1}. Dividing 1.0 by 5 equals .2, which tells us how far apart each quantified value is from the other.

So, to get good even shading, we want to take values that are 0.16 or less and map them to 0.0; values between 0.16 and 0.33 get fixed to 0.2; colorband values between 0.33 and 0.5 get quantized to 0.4, and so on up to values between 0.83 and 1.0 get mapped to 1.0.

Function f(x)
An algebraic function is made up of primitive mathematical operations (add, subtract, multiply, sine, cosine, etc) that operate on an input value to provide a desired output value.



Input		Function				Output	
8-bit	R value	*n	- ½	round()	/(n-1)	R value	8-bit
0	0.00	0.00	-0.50	0	0	0	0
13	0.05	0.30	-0.20	0	0	0	0
26	0.10	0.60	0.10	0	0	0	0
38	0.15	0.90	0.40	0	0	0	0
51	0.20	1.20	0.70	1	0.2	0.2	51
64	0.25	1.50	1.00	1	0.2	0.2	51
77	0.30	1.80	1.30	1	0.2	0.2	51
89	0.35	2.10	1.60	2	0.4	0.4	102
102	0.40	2.40	1.90	2	0.4	0.4	102
115	0.45	2.70	2.20	2	0.4	0.4	102
128	0.50	3.00	2.50	2	0.4	0.4	102
140	0.55	3.30	2.80	3	0.6	0.6	153
153	0.60	3.60	3.10	3	0.6	0.6	153
166	0.65	3.90	3.40	3	0.6	0.6	153
179	0.70	4.20	3.70	4	0.8	0.8	204
191	0.75	4.50	4.00	4	0.8	0.8	204
204	0.80	4.80	4.30	4	0.8	0.8	204
217	0.85	5.10	4.60	5	1	1	255
230	0.90	5.40	4.90	5	1	1	255
242	0.95	5.70	5.20	5	1	1	255
255	1.00	6.00	5.50	5	1	1	255

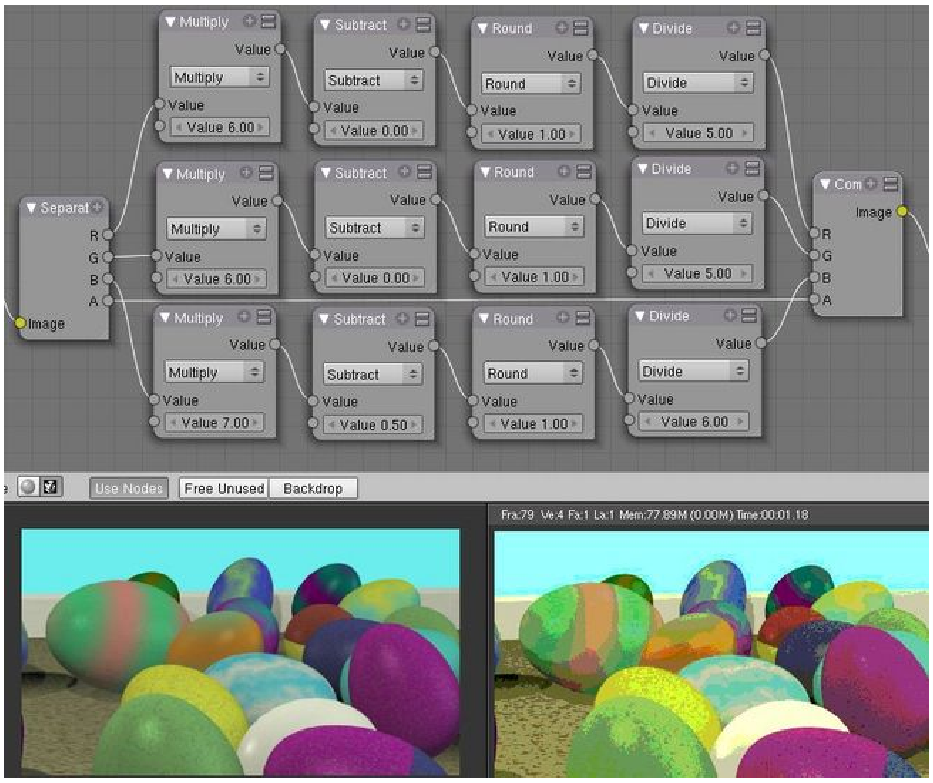
The theory behind this function is scaled truncation. Let us suppose we want a math function that takes in a range of values between 0 and 1, such as .552, but only outputs a value of 0.0, 0.2, 0.4, etc. We can imagine then that we need to get that range 0 to 1 powered up to something 0 to 6 so that we can chop off and make it a whole number. So, with six divisions, how can we do that? The answer is we multiply the range by 6. The output of that first math multiply node is a range of values between 0 and 6. To get even divisions, because we are using the rounding function (see documentation above), we want any number plus or minus around a whole number will get rounded to that number. So, we subtract a half, which shifts everything over. The Round() function then makes that range 0 to 5. We then divide by 5 to get back a range of numbers between 0 and 1 which can then be combined back with the other color channels. Thus, you get the the function

f(x,n)=round[x*n-1/2] /(n-1)

where n is the number of possible output values, and x is the input pixel color and f(x,n) is the output value. There's only one slight problem, and that is for the value exactly equal to 1, the formula result is 1.2, which is an invalid value. This is because the round function is actually a roundup function, and exactly 5.5 is rounded up to 6. So, by subtracting .501, we compensate and thus 5.499 is rounded to 5. At the other end of the spectrum, pure black, or 0, when .501 subtracted, rounds up to 0 since the Round() function does not return a negative number.

Sometimes using a spreadsheet can help you figure out how to put these nodes together to get the result that you want. Stepping you through the formula for n=6 and x=0.70, locate the line on the spreadsheet that has the 8-bit value 179 and R value 0.7. Multiplying by 6 gives 4.2. Subtracting 1/2 gives 3.7, which rounds up to 4. 4 divided by 5 = .8. Thus, f(0.7, 6) = 0.8 or an 8-bit value of 204. You can see that this same 8-bit value is output for a range of input values. Yeah! Geeks Rule! This is how you program Blender to do compositing based on Algebra. Thank a Teacher if you understand this.

Reality



To implement this function in Blender, consider the noodle above. First, feed the image to the Separate RGB node. For the Red channel, we string the math nodes into a function that takes each red color, multiplies (scales) it up by the desired number of divisions (6), offsets it by 0.5, rounds the value to the nearest whole number, and then divides the image pixel color by 5. So, the transformation is {0..1} becomes {0..6}, subtracting centers the medians to {-0.5...5.5} and the rounding to the nearest whole number produces {0,1,2,3,4,5} since the function rounds down, and then dividing by five results in six values {0.0,0.2,0.4,0.6,0.8,1.0}.

The result is that the output value can only be one of a certain set of values, stair-stepped because of the rounding function of the math node noodle. Copying this one channel to operate on Green and Blue gives the noodle below. To get the 6:6:7, we set the three multiply nodes to {6,6,7} and the divide nodes to {5,5,6}.

If you make this into a node group, you can easily re-use this setup from project to project. When you do, consider using a math node to drive the different values that you would have to otherwise set manually, just to error-proof your work.

Summary

Normally, an output render consists of 32- or 24-bit color depth, and each pixel can be one of millions of possible colors. This noodle example takes each of the Red, Green and Blue channels and normalizes them to one of a few values. When all three channels are combined back together, each color can only be one of 256 possible values.

While this example uses the Separate/Combine RGB to create distinct colors, other Separate/Combine nodes can be used as well. If using the YUV values, remember that U and V vary between -0.5 and +0.5, so you will have to first add on a half to bring the range between 0 and 1, and then after dividing, subtract a half to bring in back into standard range.

The JPG or PNG image format will store each of the colors according to their image standard for color depth (e.g. JPG is 24-bit), but the image will be very very small, since reducing color depth and quantizing colors is essentially what the JPEG compression algorithm accomplishes.

You do not have to reduce the color depth of each channel evenly. For example, if blue was the dominant color in an image, to preserve image quality, you could reduce Red to 2 values, Green to 4, and let the blue take on 256/(2*4) or 32 values. If using the HSV, you could reduce the Saturation and Value to 2 values (0 or 1.0) by Multiply by 2 and Divide by 2, and restrict the Hue to 64 possible values.

You can use this noodle to quantize any channel; alpha, speed (vector), z-values, and so forth.

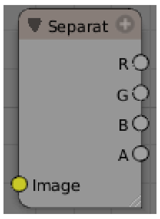
Combine/Separate Nodes

All of these node do essentially the same thing: they split out an image into (or recombine an image from) its composite color channels. Each format supports the Alpha (transparency) channel. The standard way of representing color in an image is called a *color space*. There are several color spaces supported:

- RGB: Red-Green-Blue traditional primary colors, also broadcast directly to most computer monitors
- HSV: Three values, often considered as more intuitive than the RGB system (nearly only used on computers):
  - Hue: the **Hue** of the color (in some way, choose a 'color' of the rainbow);
  - Saturation: the **quantity** of hue in the color (from desaturate - shade of gray - to saturate - brighter colors);
  - Value: the **luminosity** of the color (from 'no light' - black - to 'full light' - 'full' color, or white if Saturation is 0.0).
- YUV: Luminance-Chrominance standard used in broadcasting analog PAL (European) video.
- YCbCr: Luminance-ChannelBlue-ChannelRed Component video for digital broadcast use, whose standards have been updated for HDTV and commonly referred to as the HDMI format for component video.

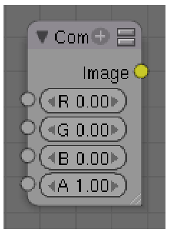
See the global wikipedia for more information on color spaces.

Separate/Combine RGBA Node



Separate
RGBA node

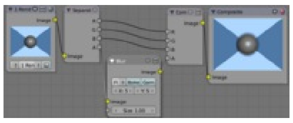
This node separates an image into its red, green, blue and alpha channels. There's a socket for each channel on the right.



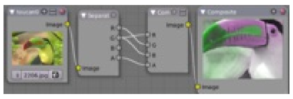
Combine
RGBAnode

This node combines separate input images as each color and alpha channel, producing a composite image. You use this node combine the channels after working on each color channel separately.

Examples

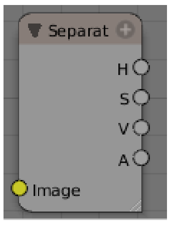


In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like anti-aliasing, but in a three-dimensional sense. Use this noodle when adding CG elements to live action to remove any hard edges. Animating this effect over a broader scale will make the object appear to "phase" in and out, as a "out-of-phase" time-traveling sync effect.



In this fun little noodle we make all the reds become green, and all the green both Red and Blue, and remove Blue from the image completely. Very cute. Very fun.

Separate/Combine HSVA Nodes

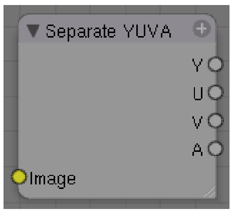


Separate HSVA
node

This node separates an image into image maps for the hue, saturation, value and alpha channels.

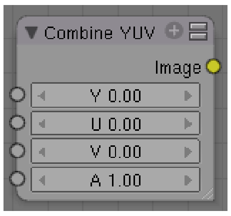
Use and manipulate the separated channels for different purposes; i.e. to achieve some compositing/color adjustment result. For example, you could expand the Value channel (by using the multiply node) to make all the colors brighter. You could make an image more relaxed by diminishing (via the divide or map value node) the Saturation channel. You could isolate a specific range of colors (by clipping the Hue channel via the Colorramp node) and change their color (by the Add/Subtract mix node).

Separate/Combine YUVA Node



Separate YUVA node

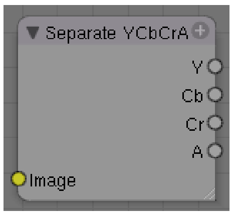
This node converts an RGBA image to YUVA color space, then splits each channel out to its own output so that they can be manipulated independently. Note that U and V values range from -0.5 to +0.5.



Combine YUVA node

Combines the channels back into a composite image. If you do not connect any input socket, you can set a default value for the whole image for that channel using the numeric controls shown.

Separate/Combine YCbCrA Node

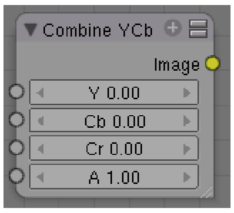


Separate YCbCrA node

This node converts an RGBA image to YCbCrA color space, then splits each channel out to its own output so that they can be manipulated independently:

- Y: Luminance, 0=black, 1=white
- Cb: Chrominance Blue, 0=Blue, 1=Yellow
- Cr: Chrominance Red, 0=Red, 1=Yellow

Note: If running these channels through a ColorRamp to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives high-contrast effect.



Combine YCbCrA node

So, I kinda think you get the idea, and I was trying to think of some other creative way to write down the same thing, but I can't. So, you'll have to figure this node out on your own.

Alpha Convert

...

Composite Matte Nodes

These nodes give you the essential tools for working with blue-screen or green-screen footage, where live action is shot in front of a blue or green backdrop for replacement by a matte painting or virtual background.

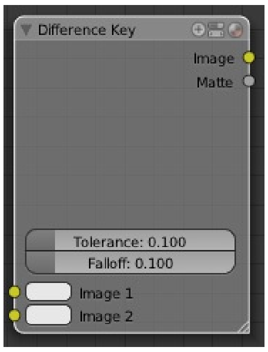
In general, hook up these nodes to a viewer, set your UV/Image Editor to show the viewer node, and play with the sliders in real-time using a sample image from the footage, to get the settings right. In some cases, small adjustments can eliminate artifacts or foreground image degredation. For example, taking out too much green can result in foreground actors looking 'flat' or blueish/purplish.

You can and should chain these nodes together, refining your color correction in successive refinements, using each node's strengths to operate on the previous node's output. There is no "one stop shopping" or one "does-it-all" node; they work best in combination.

Usually, green screen is shot in a stage with consistent lighting from shot to shot, so the same settings will work across multiple shots of raw footage. Footage shot outside under varying lighting conditions (and wind blowing the background) will complicate matters and mandate lower falloff values.

Garbage Matte
Garbage matte is not a node, but a technique where the foreground is outlined using a closed curve (bezier or nurbs). Only the area within the curve is processed using these matte nodes; everything else is garbage and thus discarded.

Difference Key Node



Difference Key node

The difference key node determines how different each channel is from the given key in the selected color space. If the differences are below a user defined threshold then the pixel is considered transparent. Difference matting does not rely on a certain background color, but can have less than optimal results if there is a significant amount of background color in the foreground object.

There are two inputs to this node.

- The first is an input Image that is to be keyed.
- The Key Color can be input as an RGB value or selected using the color picker by clicking on the Key Color box to bring up the color dialog, then clicking on the eye dropper tool and selecting a color.

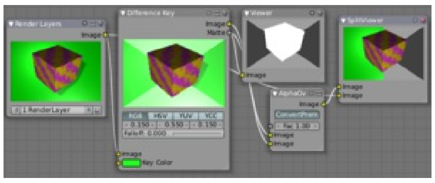
The selectable color spaces are RGB (default), HSV, YUV, and YCbCr.

You can adjust the tolerance of each color in the colorspace individually so that you can have more red variance or blue variance in what you would allow to be transparent. I find that about 0.15 (or 15%) is plenty of variance if the background is evenly lit. Any more unevenness and you risk cutting into the foreground image.

When the Falloff value is high, pixels that are close to the Key Color are more transparent than pixels that are not as close to the Key Color (but still considered close enough to be keyed). When the Falloff value is low, it does not matter how close the pixel color (Image) is to the Key Color, it is transparent.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

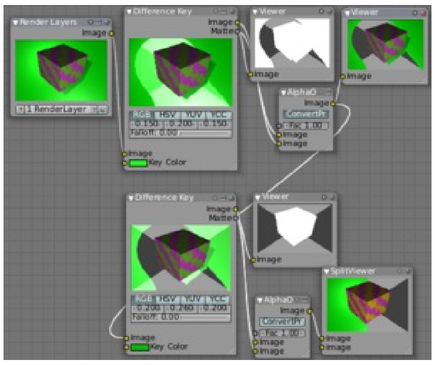
Simple Example


Using the Difference Key Node

In the example to the right (click to expand), we have a purple cube with yellow marbeling in front of a very unevenly lit green screen. We start building our noodle by threading the image to a difference key, and using the eyedropper, pick a key color very close to the edge of the cube, around where the halo is at the corner on the left-hand side; a fairly bright green. We thread two viewers from the output sockets so we can see what (if anything) the node is doing. We add an AlphaOver node, threading the Matte to the **TOP** socket and the image to the **BOTTOM** socket. Very Important, because 0 time blue is not the same as blue times zero. You always want your mask to go to the top socket of the AlphaOver. Premultiply is set and a full multiply is on so that we completely remove the green. In this example, we thread the output of the alphaover to a SplitViewer node so we can compare our results; the original is threaded to the bottom input of the SplitViewer, so that original is on the left, processed is on the right.

We set our variance to .15, and see what we get. What we get (not shown) is a matte that masks around the cube, but not on the right and around the edges where the green is darker; that shade it is too far away from our key color. So, since it is the green that is varying that we want to remove, we increase the Green variation to 1.00 (not shown). Whoa! All the Green disappears (all green within a 100% variation of our green key color is *all* the green), along with the top of the box! Not good. So, we start decreasing the green until we settle on 55% (shown).

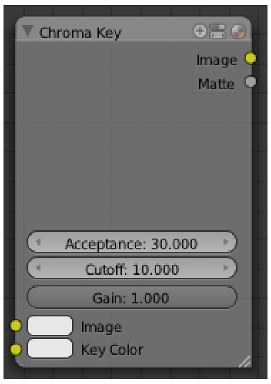
Chaining Example


Chaining Difference Key Nodes

We pay out the wazoo for our highly talented (and egotistical I might add) Mr. Cube to come into the studio and do a few takes. We told him NOT to wear a green tie, but when we look at our footage, lo and behold, there he is with a green striped tie on. When we use our simple noodle, the green stripes on his tie go alpha, and the beach background shows through. So, we call him up and, too late, he's on his way back to Santa Monica and it wasn't in his contract and it wasn't his fault, after all, we're supposed to have all this fancy postpro software yada yada and he hangs up. Geez, these actors.

So, we chain two Difference Key nodes as shown to the right, and problem solved. What we did was lower the variation percentage on the first to remove some of the green, then threaded that to a second (lower) difference key, where we sampled the green more toward the shadow side and outside edge. By keeping both variations low, none of the green in his tie is affected; that shade is outside the key's +/- variation tolerances.

Chroma Key Node



Chroma Key node

The Chroma Key node determines if a pixel is foreground or background (and thereby should be transparent) based on its chroma values. This is useful for compositing images that have been shot in front of a green or blue screen.

There is one input to this node, the Image that is to be keyed.

Control this node using:

Green / Blue buttons
: Basic selection of what color the background is supposed to be.

Cb Slope and Cr Slope (chroma channel) sliders
: Determines how quickly the processed pixel values go from background to foreground, much like falloff.

Cb Pos and Cr Pos sliders
: Determines where the processing transition point for foreground and background is in the respective channel.

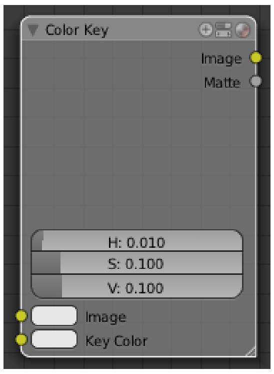
Threshold
: Determines if additional detail is added to the pixel if it is transparent. This is useful for pulling shadows from an image even if they are in the green screen area.

Alpha threshold
: The setting that determines the tolerance of pixels that should be considered transparent after they have been processed. A low

value means that only pixels that are considered totally transparent will be transparent, a high value means that pixels that are mostly transparent will be considered transparent.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

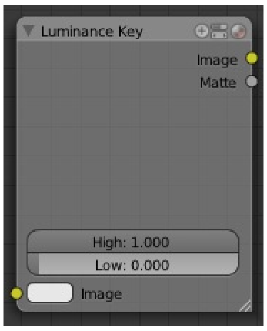
Color Key



Color Key node

The color key node creates a matte based on a specified color of the input image. The sliders represent threshold values for Hue, Saturation, and Value. Higher values in this node's context mean a wider range of colors from the specified will be added to the matte.

Luminance Key Node



Luminance Key node

The Luminance Key node determines background objects from foreground objects by the difference in the luminance (brightness) levels. For example, this is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.

There is one input to this node, the Image that is to be keyed.

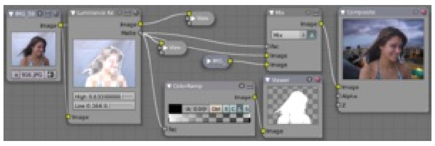
Control this node using:

- The High value selector determines the lowest values that are considered foreground. (which is supposed to be - relatively - light: from this value to 1.0).
- The Low value selector determines the hightes values that are considered to be background objects. (which is supposed to be - relatively - dark: from 0.0 to this value).

It is possible to have a separation between the two values to allow for a gradient of transparency between foreground and background objects.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

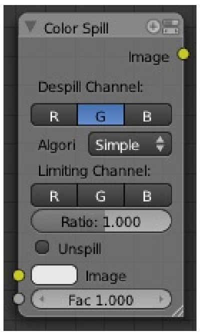
Example


Using Luma Key...with a twist

For this example, let's throw you a ringer. Here, the model was shot against a *white* background. Using the Luminance Key node, we get a matte out where the background is white, and the model is black; the opposite of what we want. If we wanted to use the matte, we have to switch the white and the black. How to do this? ColorRamp to the rescue - we set the left color White Alpha 1.0, and the right color to be Black Alpha 0.0. Thus, when the Colorramp gets in black, it spits out white, and vice versa. The reversed mask is shown; her white outline is useable as an alpha mask now.

Now to mix, we don't really need the AlphaOver node; we can just use the mask as our Factor input. In this kinda weird case, we can use the matte directly; we just switch the input nodes. As you can see, since the matte is white (1.0) where we don't want to use the model picture, we feed the background photo to the bottom socket (recall the mix node uses the top socket where the factor is 0.0, and the bottom socket where the factor is 1.0). Feeding our original photo into the top socket means it will be used where the Luminance Key node has spit out Black. Voila, our model is teleported from Atlanta to aboard a cruise ship docked in Miami.

Color Spill Node



Color Spill node

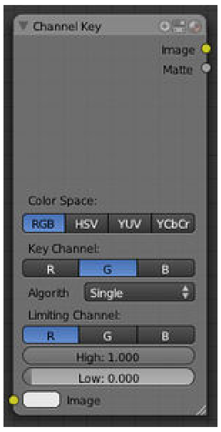
The Color Spill node reduces one of the RGB channels so that it is not greater than any of the others. This is common when compositing images that were shot in front of a green or blue screen. In some cases, if the foreground object is reflective, it will show the green or blue color; that color has "spilled" onto the foreground object. If there is light from the side or back, and the foreground actor is wearing white, it is possible to get "spill" green (or blue) light from the background onto the foreground objects, coloring them with a tinge of green or blue. To remove the green (or blue) light, you use this fancy node.

There is one input to this node, the Image to be processed.

The Enhance slider allows you to reduce the selected channel's input to the image greater than the color spill algorithm normally allows. This is useful for exceptionally high amounts of color spill.

The outputs of this node are the image with the corrected channels.

Channel Key Node



Channel Key node

The Channel Key node determines background objects from foreground objects by the difference in the selected channel's levels. For example in YUV color space, this is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.

There is one input to this node, the Image that is to be keyed.

Control this node using:

- Color Space buttons selects what color space the channels will represent.
- Channel buttons selects the channel to use to determine the matte.
- High value selector determines the lowest values that are considered foreground. (which is supposed to be - relatively - height values: from this value to 1.0).
- Low value selector determines the highest values that are considered to be background objects. (which is supposed to be - relatively - low values: from 0.0 to this value).

It is possible to have a separation between the two values to allow for a gradient of transparency between foreground and background objects.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

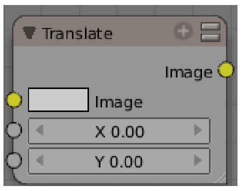
Distance Key

...

Composite Distort Nodes

These nodes distort the image in some fashion, operating either uniformly on the image, or by using a mask to vary the effect over the image.

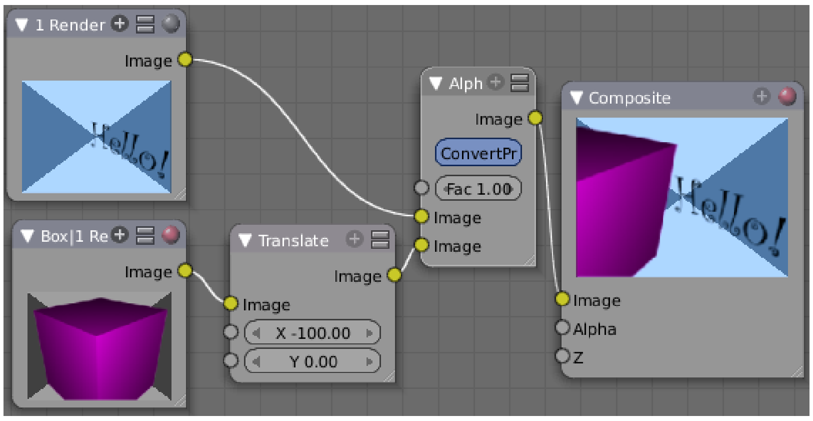
Translate Node



Translate node

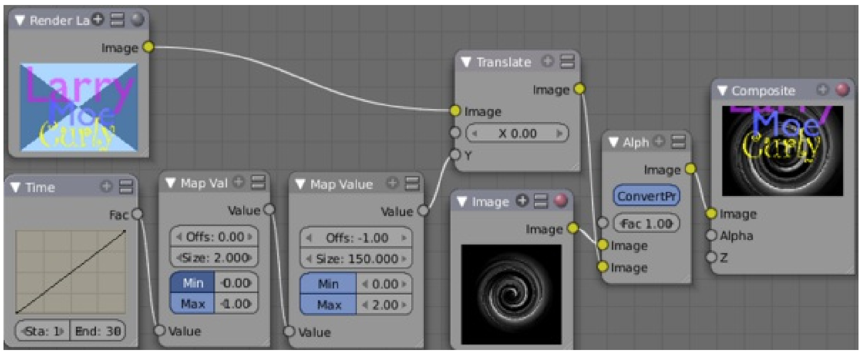
The translate node translates (moves) an image by the specified amounts in the X and Y directions. X and Y are in pixels, and can be positive or negative. To shift an image up and to the left, for example, you would specify a negative X offset and a positive Y.

Use this node to position an image into the final composite image. Usually, the output of this node is routed to an AlphaOver or Mix node to mix it with a base image. In the example below, the RenderLayer input from one scene (box) is translated over to the left (a negative X translation) and alphaovered with a Hello scene RenderLayer input



Example: Using the Translate Node to Roll Credits

At the end of your fantastic animation, you'll want to give credit where credit is due. This is called rolling the credits and you see the names of everyone involved scroll up over a background image or sequence. The mini-map below shows an example of how to roll credits using the Translate node.



Using the Translate Node to Roll Credits

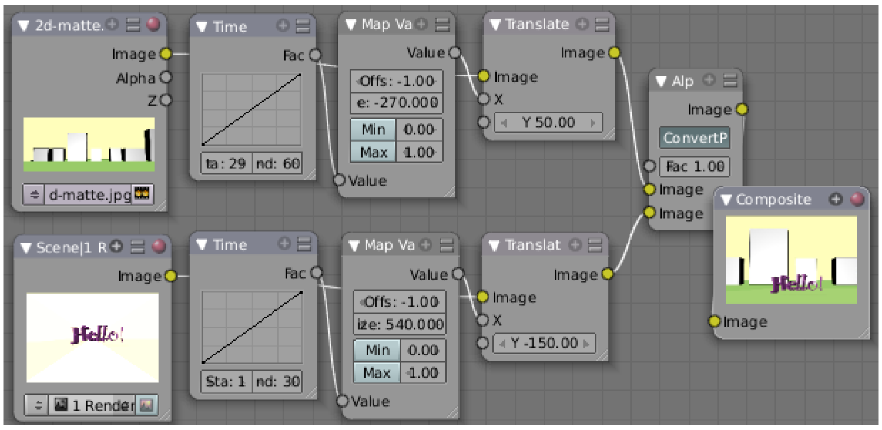
In this node map, the RenderLayer input has a list of the people involved and is 150 pixels high; it is the image input into the Translate Node. The Y value (vertical) offset of the Translate node comes from a scaled time factor that varies from -150 to 150 over 30 frames. The Translated image is overlaid on top of a background swirl image. So, over the course of 30 frames, the Translate node shifts the image from down by 150 pixels (off the bottom of the screen), up through and overlaid on top of the swirl, and finally off the screen to the top. These frames are generated when the Render Animation buttons are set and Anim is pressed. Right now, frame 21 is showing Moe and Curly, and Larry has scrolled off the screen.

Alpha channel
Be sure to save your credits image in a format that supports an alpha channel, such as PNG, so that the AlphaOver node can overlay it on the background and let the background show through.

You *could* have parented and animated all of the text to roll up past your camera, but rendering would have taken forever. Using the translate node is much faster to composite, and more flexible. To add someone to the list, simply change the credits image and re-animate. Since it is simple image manipulation, Blender is blazingly fast at regenerating your credits. Similarly, changing the background is rock simple as well; simply load up a different background image and re-Animate.

Example: Moving a Matte

In some 2D and 3D animations and movies, a matte painting is used as the background. In most scenes it is still, however you can easily move it using the Translate node. Mattes are huge; the example used below is actually 1440x600 pixels, even though the scene being rendered is for TV. This oversizing gives us room to move the matte around. The example noodle below introduces a dancing "Hello!" from stage right in frames 1-30. As the "Hello" reaches center stage, we fake a camera move by moving the matte to the left, which makes it look like the "Hello!" is still moving as the camera moves with it.



Moving a Matte in back of a moving Animation (frame 60)

Use the Map Value node to scale the X (left to right) offsets that feed the Translate node. Note that offsets are used to position the dancing "Hello!" down to look like it is walking along the street (in the render scene, it is centered on camera and just dances in place). The matte is adjusted up to fake a camera height of an observer, bringing the horizon up.

Example: Shake, Rattle and Roll

A real world effect is the shaking of the camera. BOOM! We expect to feel the impact and see it rock our world. In our virtual CG world, though, we are in the vaccuum of space. So, we have to fake a camera being shook. There are two points in the development cycle to do this: at *Render-Scene* time, and at *Composite* time.

At *Render-Scene* time, the modeler would introduce an Ipo curve and keyframes that rotate and/or move the camera for a short amount of time. The advantage to render-scene shake is that the artist handles it and the editor does not have to worry about it; one less thing to do thank goodness. The disadvantage is that the artist may only be modeling the actors, and not the background scenery, props, or matte, so any shake they introduce does not transfer to the set, props, or backdrop. Therefore, it is best to introduce camera shake after all scenes have been rendered and assembled and when they are being composited.

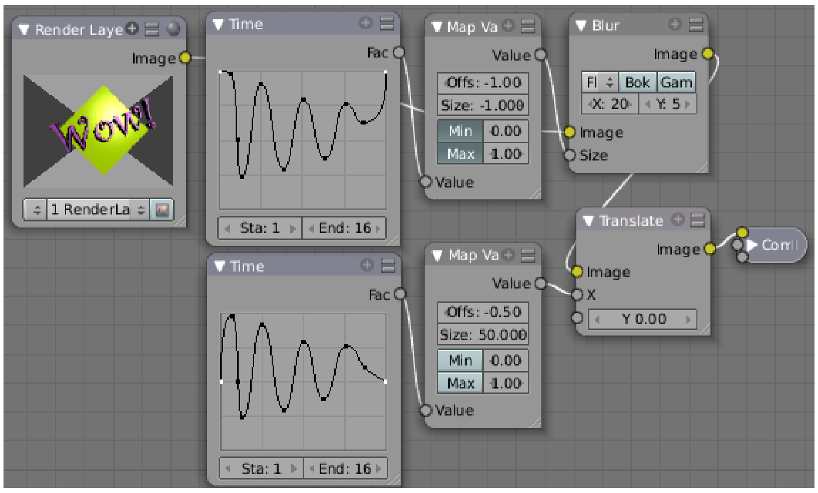
There are two aspects to being bumped, or tripping over the tripod, or having an explosion go off next to you, or an airplane have a near miss as it flies by, or throwing up on a long sea voyage, or surviving an earthquake: *camera motion* and *image blur* (I know you were thinking expletives and changing your underpants, but this is about compositing).

Camera Motion happens because the camera physically gets moved; but its mass and its tripod also acts as a dampening device, softening out and absorbing the initial bump. The cameraman also acts as a dampener, and also as a corrector, trying to get the camera back to where it was pointed originally.

There can be quite a delay between the shock and the correction; for example, a lone actor/cameraman may trip on the tripod coming out from behind the camera, come into frame, realize the camera is off, and then come back to correct it. It all depends on the artistic effect and story you want to convey.

The *image blur* comes into play because the shake happens so rapidly that the image is blurred in the direction of the shake. However, the blur is more when the camera is being pushed back into position, and less when the camera is at the extreme of its deflection, since it is decelerating at the apex of its movement. Like motion, blur is the most during the initial shock, and less as things slow down and get under control. Also, the camera may go out of focus and come back into focus at the end of the shake.

To use Blender nodes to mimic Camera Motion, use the noodle shown below. The noodle has a Blur group on top that feeds a Translate group below it.



SFX: Camera Shake

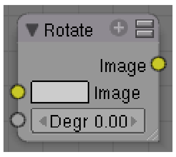
In the above example, we use a Time curve that mimics the intensity and duration of a typical BOOM!. In this case, both curves have four peaks within a 16-frame period to mimic a BOOM! (in fact one curve was constructed and then duplicated to make the other, to ensure that the bulk of both curves was exactly the same). Notice how the curve dampens (decreases in magnitude as time progresses) as discussed above. Notice how the curve slows down (increasing period) to mimic the cameraman getting it back under control. Notice that the curve is sinusoidal to mimic over-correction and vibration.

BOOM! to the Left: The translate curve starts at 0.5. Maximum deflection up is fully a half, yet down is only a quarter. This offset mimics a BOOM! off to our left, since the camera shakes more to the right, away from the BOOM!

Motion and Blur are the same but different: Notice that the two curves are identical except for the highlighted start and end dots; we want zero blur and zero offsets before and after the shake, but minimum blur when there is maximum translate. The two Map Value node settings are different to achieve this; the math is left to the reader.

Use this Blender noodle to mimic camera shake. The amount of shake is set by the Size values, and the blur should be proportional to the amount and direction of motion (predominantly X in this example). Use the Time start and end to vary the duration of the shake; ten seconds for an earthquake, one minute for a ship rolling in the seas, a half second as an F-14 flies by and takes your ear off. *Author's note: I noticed cool camera shakes while watching the Halo 3 previews.*

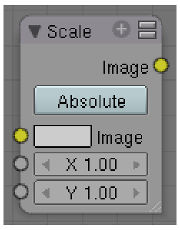
Rotate Node



Rotate node

This node rotates an image. Positive values rotate clockwise and negative ones counterclockwise.

Scale Node



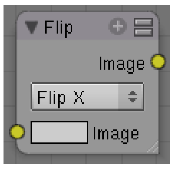
Scale node

This node scales the size of an image. Scaling can be either absolute or relative. If Absolute toggle is on, you can define the size of an image by using real pixel values. In relative mode percents are used.

For instance X: 0.50 and Y: 0.50 would produce image which width and height would be half of what they used to be. When expanding an image greatly, you might want to blur it somewhat to remove the square corners that might result. Unless of course you want that effect; in which case, ignore what I just said.

Use this node to match image sizes. Most nodes produce an image that is the same size as the image input into their top image socket. So, if you want to uniformly combine two images of different size, you must scale the second to match the resolution of the first.

Flip Node



Flip node

This node flips an image at defined axis that can be either X or Y. Also flipping can be done on both X and Y axis' simultaneously.

You can use this node to just flip or use it as a part of mirror setting. Mix half of the image to be mirrored with its flipped version to produce mirrored image.

Displace Node

Ever look down the road on a hot summer day? See how the image is distorted by the hot air? That's because the light is being bent by the air; the air itself is acting like a lens. This fancy little node does the same thing; it moves an input image's pixels based on an input vector mask (the vector mask mimics the effect of the hot air).

This can be useful for a lot of things, like hot air distortion, quick-and-dirty refraction, compositing live footage behind refracting objects like looking through bent glass or glass blocks, and more! Remember what HAL saw in 2001:Space Odyssey; that distorted wide-angle lens? Yup, this node can take a flat image and apply a mask to produce that image.

The amount of displacement in the X and Y directions is determined by

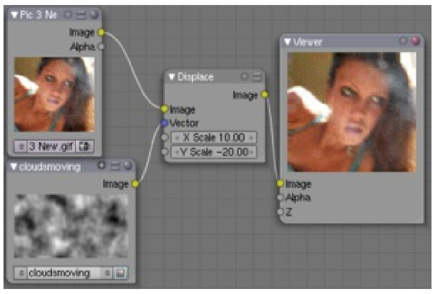
- The value of the mask's channels:
- The scaling of the mask's channels

The (red) channel 1's value determines displacement along the positive or negative X axis. The (green) channel 2's value determines displacement along the positive or negative Y axis.

If both the channels' values are equal (i.e. a greyscale image), the input image will be displaced equally in both X and Y directions, and also according to the X scale and Y scale buttons. These scale button act as multipliers to increase or decrease the strength of the displacement along their respective axes. They need to be set to non-zero values for the node to have any effect.

Because of this, you can use the displace node in two ways, with a greyscale mask (easy to paint, or take from a procedural texture), or with a vector channel or RGB image, such as a normal pass, which will displace the pixels based on the normal direction.

Example



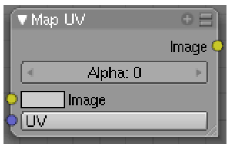
Music Video Distortion Example Using Displace

In this example, she's singing about dreams of the future. So, to represent this, we use a moving clouds texture (shot just by rendering the cloud texture on a moving plane) as the displacement map. Now, the colors in a black and white image go from zero (black) to one (white), which, if fed directly without scaling would only shift the pixels one position. So, we scale their effect in the X and Y direction.

Upon reviewing it, sometimes stretching in both the X and Y direction made her face look fat, and we all can guess her reaction to looking fat on camera. SO, we scale it only half as much in the X so her face looks longer and thinner. Now, a single image does not do justice to the animation effect as the cloud moves, and this simple noodle does not reflect using blur and overlays to enhance (and complicate) the effect, but this is the core.

Photos courtesy of Becca, no rights reserved. See also some movies of this node in action, made by the wizard programmer himself, by following this external link

Map UV Node



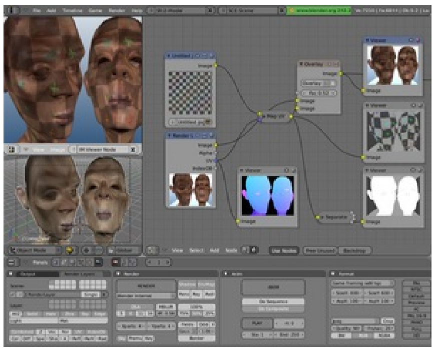
So, I think we all agree that the problem is...we just don't know what we want. The same is true for directors. Despite our best job texturing our models, in post production, inevitably the director changes their mind. "Man, I really wish he looked more ragged. Who did makeup, anyway?" comes the remark. While you can do quite a bit of coloring in post production, there are limits. Well, now this little node comes along and you have the power to **re-texture your objects *after* they have been rendered**. Yes, you read that right; it's not a typo and I'm not crazy. At least, not today.

Using this node (and having saved the UV map in a multilayer OpenEXR format image sequence), you can apply new flat image textures to all objects (or individual objects if you used the very cool ID Mask Node to enumerate your objects) in the scene.

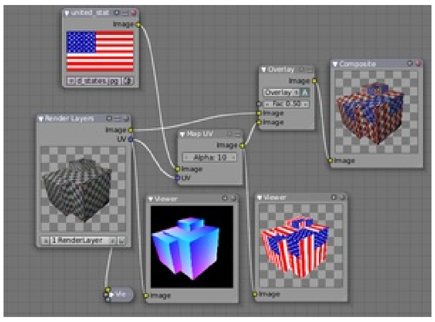
Thread the new UV Texture to the Image socket, and the UV Map from the rendered scene to the UV input socket. The resulting image is the input image texture distorted to match the UV coordinates. That image can then be overlay mixed with the original image to paint the texture on top of the original. Adjust alpha and the mix factor to control how much the new texture overlays the old.

Of course, when painting the new texture, it helps to have the UV maps for the original objects in the scene, so keep those UV texture outlines around even after all shooting is done.

Examples


Adding a Grid UV Textures for Motion Tracking

In the example to the right, we have overlaid a grid pattern on top of the two Emo heads after they have been rendered. During rendering, we enabled the UV layer in the RenderLayer tab (Buttons window, Render Context, RenderLayer tab). Using a mix node, we mix that new UV Texture over the original face. We can use this grid texture to help in any motion tracking that we need to do.


Adding UV Textures in Post-Production

In this example, we overlay a flag on top of a cubie-type thing, and we ensure that we Enable the Alpha pre-multiply button on the Mix node. The flag is used as additional UV Texture on top of the grid. Other examples include the possibility that we used an unauthorized product box during our initial animation, and we need to substitute in a different product sponsor after rendering.

Of course, this node does NOT give directors the power to rush pre-production rendering under the guise of "we'll fix it later", so maybe you don't want to tell them about this node. Let's keep it to ourselves for now.

Crop Node

The Crop Node takes an input image and crops it to a selected region.

Crop Image Size
: When enabled, the image size is cropped to the specified region. When disabled, image remains the same size, and uncropped areas become transparent pixels.

Relative
: When enabled, crop dimensions are a percentage of the image's width and height. When disabled, the range of the sliders are the width and height of the image in pixels.

Crop Region Values
: These sliders define the lower, upper, left, and right borders if the crop region.

Lens Distortion

Use this node to simulate distortions that real camera lenses produce.

Distort
: This creates a bulging or pinching effect from the center of the image.

Dispersion
: This simulates chromatic aberration, where different wavelengths of light refract slightly differently, creating a rainbow colored fringe.

Projector
: Enable or disable slider projection mode. When on, distortion is only applied horizontally. Disables Jitter and Fit.

Jitter
: Adds jitter to the distortion. Faster, but noisier.

Fit
: Scales image so black areas are not visible. Only works for positive distortion.

Sequence Editing

In addition to modeling and animation, Blender has a fully functional Video Sequence Editor (VSE) as well as an advanced node-based editor that also manipulates a video stream. Compositing Nodes operate equally well on images or video streams, and can apply detailed image manipulation on the stream.

Operating at a higher conceptual level, and used later in the video production process, Blender's legacy VSE operates on a set of entire strips at a time, as a chunk of footage. The many parts of Blender work together in typical work flow fashion:

- Model to construct the objects
- Assign Materials and introduce Lighting to color the objects
- Animate your objects to make them move
- Render layers of video using cameras
- Use Compositing nodes to:
    - Enhance the images by adjusting colors, adding in-scene special effects
    - Layer the images into a composite image sequence (strip)
- Assemble the video strips together to make a movie using the VSE.

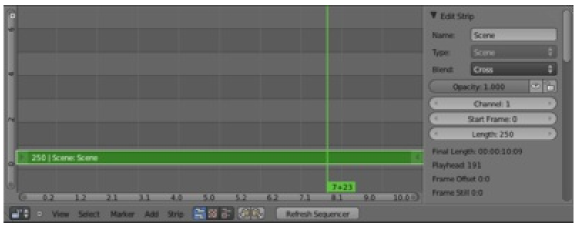
The VSE within Blender is a complete video editing system that allows you to combine multiple video channels and add effects to them. Its functionality has been inside Blender since the beginning. Even though it has a limited number of operations, you can use these to create powerful video edits (especially when you combine it with the animation power of Blender!) Furthermore, it is extensible via a plugin system to perform an unlimited number of image manipulations.

Using the VSE, you load multiple video clips and lay them end-to-end (or in some cases, overlay them), inserting fades and transitions to link the end of one clip to the beginning of another. Finally, add an audio track so you can synchronize the timing of the video sequence to match it. The result of using the VSE is your finished movie.

FFMPEG Support
Support for exporting an avi/quicktime movie using FFMPEG does work, currently (since 2.44) only within the Linux and Windows builds. With FFMPEG support, you are able to save the audio track with your video.

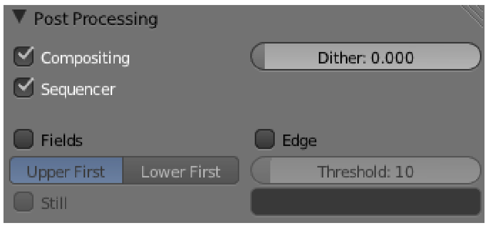
Overview of the Sequence Editor

Blender's Video Sequence Editor is a flexible workbench for editing your video footage. It is used to review your footage, and glue many sequences of your movie together. It offers a number of built-in and plug-in effects to transition from sequence to sequence, providing advanced hollywood-style effects for a professional looking video.


Video Sequence Editor in Sequence display mode

The Video Sequence Editor has a header (where the menu and view modes are shown) and a workspace, and works in one of several view modes. The Marker menu allows you to add markers in the VSE. Markers are shared across animation editors. See Markers

The sequencer workspace is horizontally striped into channels and each video strip will go in a horizontal channel. Each channel is numbered on the left-hand side, starting from 0 (you can't put anything thing in this special one!) and going up. Stripes toward the bottom are more dominant, which we'll get to in a minute. In the x direction, seconds of animation or frames of animation (CtrlT to choose) are used as the measure of time (seconds 1 through 7 are shown). You can scale the time using the zoom keys or mouse actions (see the Reference for more info).



Sequence Output Enabled

When you click Render or Anim to generate an image or video, Blender has a choice of what image to compose for the current frame/scrub range:

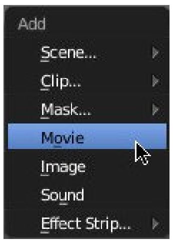
- Current Scene layer result
- Sequence Editor channel 0 result
- Composition Node Editor renderlayer result

The video sequencer is enabled by default.

When you go to the render panel where ordinary renderings take place and you click animation or image with the VSE open, it will render the clips for the VSE instead of the scene.

Using the Sequence Editor

Adding Strips



the add menu

The Add menu is the main menu you will be using to add content to the VSE. In general, you load up your strips, create strips of special transition effects, and then animate out your sequence by selecting "Do Sequence" and clicking the Anim button. You can use the Add menu in the header, or hover your mouse cursor over the Sequence workspace and press ⇧ ShiftA.

Clips can be Huge
A three minute quicktime .mov file can be 140Megs. Loading it, even over a high-speed LAN can take some time. Don't assume your computer or Blender has locked up if nothing happens for awhile.

First, let's add a clip:

- A movie clip in the Audio-Video Interleaved format (*.avi file)
- A movie clip in the Apple QuickTime format (*.mov)

- A single still image to be repeated for a number of frames (*.jpg, *.png, etc.)
- A numbered sequence of images (*-0001.jpg, *-0002.jpg, *-0003.jpg, etc, of any image format)
- One or more images from a directory
- A Scene in your .blend file.

Blender does not care which of these you use; you can freely mix and match any of them. They all become a color-coded strip in the VSE:

- Blue is used for Avi/mov codec strips
- Grey is a single image that is repeated/copied
- Purple is an image sequences or group of images played one after the other
- Green is an Audio track

When you choose to add one of these, the VSE window will switch to a file browser for you to select what you want to add. Supported files have a little rectangle next to their name (blue for images, green for clips) as a visual cue that you can pick them successfully:

Add Movies or Images

When adding a Movie or Movie+Audio LMB LEFT CLICK to put the name of the file into the text box at the top; this selects a **single** file (like a movie)

In the case of (numbered) image **sequences**, you have a choice: **Directory**: RMB right-click on a directory name, and all files in that directory will be brought in as part of the image, in sort order, one image per frame **Range**: Navigate into the directory and right-click and drag over a range of names to highlight multiple files. You can page down and continue right-click-dragging to add more to the selection **Batch**: Shift-right-click selected non-related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (jpg, png, exr, etc.) **All**: Press A to select/deselect All files in the directory.

When you click the Select <whatever> button, the window pane will switch back to VSE, and the strip will be rubber-banded to your mouse. You cannot load multiple movies at the same time by right-clicking them; no movies load if you right click them. Right-clicking only works for images.

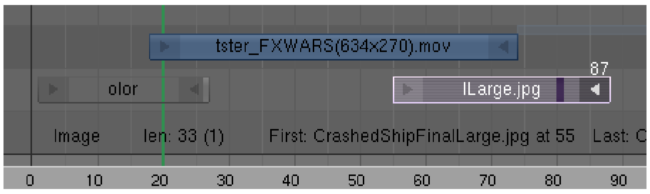
Error: The selected file is not a movie or FFMPEG support not compiled in!
means that the file is not a movie that Blender can recognize, or **you selected with the wrong button**. You get this error message because you *right*-clicked on a movie file, OR you don't have a codec that can decode the avi file. If it's the latter, find a codec so you can play the file outside of Blender, and then you will be able to load it. If it's the former, you must left-click to select movies.

In order to add items to the VSE, left-click for movies, left-click for single images, or right-click and drag for image sequences. Move your mouse to the frame/time and stripe you want, and click to break the rubberband and drop the strip in place (in a channel and starting at a frame).

When you add an image, Blender makes it into a 50-frame strip, which means that image will be in your video for two seconds (at 25 fps – PAL). Aside from re-positioning it, you will want to scale it by RMB -clicking on either the start or end arrow, and dragging left or right. As you move, the frame number updates to say where the arrow is. Click LMB to validate, or RMB to cancel the modification.

Dealing with Different Sizes

Dealing with different sized images and different sized outputs is tricky. Think like a pixel. If you have a mis-match between the size of the input image and the render output size, the VSE does try to auto-scale the image to fit it entirely in the output. This may result in clipping. If you do not want that, use Crop and/or Offset in the Input panel to move and select a region of the image within the output. When you use Crop or Offset, the auto-scaling will be disabled and you can manually re-scale by adding the Transform effect.


tster_FXWARS(634x270).mov
olor
87
lLarge.jpg
Image
len: 33 (1)
First: CrashedShipFinalLarge.jpg at 55
Last: C
0
10
20
30
40
50
60
70
80
90

If you scroll up the workspace, you will see an information channel (at vertical location channel 0) that gives you some helpful hints about the active strip. The example above shows a color strip from frames 1 to 25, then a mov file, and then an image strip. The info channel shows handy information about the image strip, whose name has been scrunched in the strip display, but is clearly spelled out in the information strip.

9999 frames go by (IMAGE strips only!)
Ok, so that was a very obscure reference to a song about 99 balloons, but we really have not anticipated how fast Blender has moved into mainstream video editing. Unfortunately, we initially reserved 4 digits for the filename of each video image sequence set. While that provides for up to 400 seconds of video (about 5 minutes US), with Blender moving into movies, you need to break up IMAGE strips into 4 digits only, and others 5 digits (10000-19999), (20000-29999), etc. Important: that only affects IMAGE strips at the moment. All the other strip types work fine with up to 300,000 frames (approximately 3 hours, read: Ben Hur :)).

Codecs
You must have a codec on your machine that can decode the avi file. Blender does not control these. For example, the XviD codec is available from www.xvid.org

FFMPEG Support
If you are using a Blender build with FFMPEG support, you will be able to load audio and video strips together; select Movie+Audio(HD) and when you drop the strip, the strip will split into an audio and video channel strips.

Add Scene

You can add the virtual image output of a Scene in your current .blend file as well. Select the scene from the popup list, and a strip will be added and rubberbanded to your mouse just like a movie or image. The strip length will be determined based on the animation settings in that scene (not the current scene, unless the VSE is operating in the same scene).

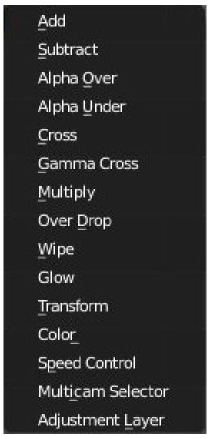
When adding a Scene strip, please note that, in order to show you the strip in the VSE Image preview mode, Blender must render the scene. This may take awhile if the scene is complex, so there may be a delay between the time you select the scene and the time the strip appears. To reduce the delay, simplify the scene rendering by selecting fewer layers to render.

If the extra overhead of rendering the scene becomes burdensome (for either preview or for multiple test renders) and you have enough disk space consider rendering the scene to a sequence of PNGs and using an Image Sequence strip instead of a scene. This is very popular for static graphic overlays like title cards which are often little more than a static image with animated opacity.

Add Audio

The VSE can incorporate an audio channel which you can hear as you scrub. Add an audio track when you are trying to time your video/animation to an audio track, or vice versa. Please refer to the Audio Sequences section for more information.

Adding Effects



Available Built-in Effects

Blender offers two categories of effects: Built-in and Plug-in. The built-in effects are listed to the right. They are built-in to Blender and everyone has them. The plug-in effects are separate files in a sequence-plugin directory on your PC that are loaded as they are needed. While a standard set of plugins are distributed when you installed Blender, everyone's computer may have a different set.

Every Built-in effect is explained in the next page individually, but they all are added and controlled in the same way. To add an effect strip, select one base strip (image, movie, or scene) by RMB clicking on it. For some effects, like the Cross transition effect, you will need to ⇧ Shift RMB a second overlapping strip (it depends on the effect you want). Then select Add -> Effect and pick the effect you want from the pop-up menu. When you do, the Effect strip will be shown above the source strips. If it is an independent effect, like the color generator (described later), it will be rubberbanded to your mouse; click to drop the strip.

Since most Effects strips depend on one or two source strips, their frame location and duration depends on their source strips. Thus, you may not be able to move it; you have to move the source strips in order to affect the effect strip.

To use an effect that combines or makes a transition between (or composites) two strips, you must Box select or shift-right-click two of them. When you add the effect strip, it will be placed in a channel above the two in Grab mode (click to drop it on a channel). Its duration will be the overlap between the two strips as a maximum.

With some effects, like the AlphaOver, the order in which you select the strips is important. You can also use one effect strip as the input or source strip with another strip, thus layering effects on top of one another.

Note: The only exception is the Color Generator effect. It does not depend on a base strip; you can add and position it independent of any other strip. Change the length as you would any strip.

Mode: Sequence, Effects Strip Selected

Hotkey: C

Menu: Strip -> Change Effect

If you picked the wrong effect from the menu, you can always change it by selecting the strip (RMB) and using the Strip->Change Effect selection. Or, you can press Change to switch effects on a selected Effects strip.

Adding Plugin Effects

Not Yet Implemented

VSE plugins are not working in Blender 2.6 currently…

Strip Properties

The properties for the strip are examined and set in the properties panel, shortcut N.

- Edit Strip - change properties of the strip
- Strip Input - where to pull images from
- Effect - Settings for effects strips
- Filter - Image pre-processing
- Proxy - Use representatives of the real image, for low-powered PCs
- Scene - Settings for when a scene strip is selected
- Sound - Settings for a sound clip

The panels for each of these sets of options and controls are shown to the right

Edit Strip Panel

Name
: You can name or rename your strips here.

Type
: Displays the type of strip selected.

Blend Mode
: By default, a strip Replaces the output image of any lower-level strips. However, many other blending modes are available based on the strip type. For example, Alpha-Over automatically overlays the image on top of a lower level strip. Autoblending modes remove the need for separate effect strips. Blend percent controls how much of an effect the strip exerts, even over time.

Opacity
: Set the opacity of the strip.

Mute
: Hides the strip so that it does not participate in the final image computation

Lock
: Prevents the strip from being moved.

Channel
: Changes the channel number, or row, of the strip.

Start Frame
: Changes the starting frame number of the strip, which is the same as grabbing and moving the strip. Tip: when you add a strip, I like to just drop it and then use this field to place it at the frame I want, rather that trying to drag and drop in exactly the right place.

Length
: Specify the number of frames to use for the strip.

Use the Convert to Premul button if a strip has an Alpha (transparency) channel. Use FilterY if the strip is from broadcast video and has even or odd interlacing fields. Enhance the color saturation through the Multiply field. Play a strip backwards by enabling Reverse Frames. Tell Blender to display every nth frame by entering a Strobe value. Finally, when using MPEG video (VCD, DVD, XVid, DivX, …), an image is built up over the course of a few frames; use the Preseek field to tell Blender to look backward and compose the image based on the n previous frames (e.g. **15** for Mpeg2 DVD).

Effect Strip

For all effects, use the Strip Properties panel to control the effects strip; each effect has different controls, but they can all be set in the Properties panel. Control the length of the strip to vary the speed with which the transform happens. Regardless of whether they are built-in or plug-in, all effect strips do some special image manipulation, usually by operating on another strip or two in a different channel. The effect strip is shown in some channel, but its resultant effect shows up as Channel 0.

Strip Input

Controls the source of the strip. Fields include file path, file name, image offset, crop settings.

This is here you can edit/update the path of the file used by a strip. Very useful when you moved it one way or the other – this avoid you deleting and re-creating the strip!

You have two text fields for path, the first being the path of the parent directory (Path), and the second the file name itself.

Filter

Enables you to quickly set common image pre-processing options.

Strobe

Flip
: X flips (reverses) the image left-to-right, Y reverses top-to-bottom.

Backwards
: Reverses strip image sequence

De-Interlace
: Removes fields in a video file.

Saturation
: Increase or decrease the saturation of an image.

Multiply
    Multiplies the colors by this value.
Premultiply
    Premultiply the Alpha channel.
Convert Float
    Converts input to float data.

Use Color Balance
    Provides three filters to adjust coloration: Lift, Gamma, and Gain. Each pass can have a positive, or inverted effect by clicking the appropriate button. Set the amount of the effect by setting the color swatch; white (RGB 1,1,1) has no effect.

Proxy Strip Properties Panel

A proxy is a smaller image (faster to load) that stands in for the main image. When you Rebuild proxy Blender computes small images (like thumbnails) for the big images and may take some time. After computing them, though, editing functions like scrubbing and scrolling and compositing functions like cross using these proxies is much faster but gives a low-res result. Disable proxies before final rendering.

In order to actually *use* the proxies, the proper "Proxy Render Size" dropdown value must be selected in the Properties panel of the Sequencer View (where the edit plays back).

Sound

This panel appears when a sound file is selected.

Here you can specify the Sound Strip's file path and file name.

Pack
    Packs the sound file into the current .blend file.
Caching
    The sound file is decoded and loaded into RAM.
Volume
    Set the volume of the Sound file.
Attenuation/dB
    Attenuation in decibels
Trim Duration: Start/End
    Offset the start and end of a sound strip.

Scene

Specify the scene to be linked to the current scene strip.

Sequencer
    Process the render (and composited) result through the video sequence editor pipeline, if sequencer strips exist. This is the same function as in the render settings.
Camera Override
    Change the camera that will be used.

Adjusting the View

Use these shortcuts to adjust the sequence area of the VSE:

Pan MMB
Zoom Wheel
Vertical Scroll use ⇧ Shift Wheel, or drag on the left scroll bar.
Horizontal Scroll use Ctrl Wheel, or drag on the lower scroll ;bar.
Scale View Vertically, drag on the circles on the vertical scroll bar.
Scale View Horizontally, drag on the circles on the horizontal scroll bar.

As usual, the View Menu controls what and how you view in the workspace.

Properties Panel
    The Properties Panel contains options for the the way the preview is displayed.
View all Sequences ↖ Home
    Zooms (out) the display to show all strips.
Fit preview in Window ↖ Home
    Resizes preview so that it fits in the window.
Show Preview 1:1 1 NumPad
    Resizes preview to a 1:1 scale (actual size).
View Selected . NumPad
    Zooms in the display to fit only the selected strips

Use this when working arranging a lot of strips and you want to use all of your screen to work.

Mode: Sequence

Hotkey: T

Menu: View -> Show Frames, View -> Show Seconds

Draw Frames
: Diplays the frame number instead of the time, in the Frame Number Indicator.

Show Frame Number Indicator
: Toggles the units of measure across the bottom of the workspace between seconds or frames.

Safe Margin
: Displays an overlay on the preview, marking where title safe region is.

Separate Colors
: When using Luma Waveform view, this separates R,G, and B into separate graphs.

Transform Markers
: Transform Markers as well as Strips.

Scrubbing

To move back and forth through your movie, use the Timeline window. LMB click and drag left/right in the timeline window, moving the vertical bar which indicates the current frame. As you do, the image for that frame is displayed in the VSE window.

Real-time scrubbing and image display is possible on reasonable computers when viewing an image sequence or movie (avi/mov) file. Scene images have to be rendered individually, which may take some time.

View Modes

The icons in the header allow to change the view of the VSE. By default, only the sequencer is displayed. The second button displays only the Preview window, and the third button displays both the Sequencer and the Preview.

When the preview is enabled, you have several options to change what type pf preview to display. They are explained in the Display Modes Page.

Scene Preview

When using a Scene Strip in the sequencer, these settings in the Properties Panel determine how they are shown in the preview window.

Open GL Preview
: If you have Open GL, enable this setting to use Open GL for the scene preview renders.
: The drop down menu allows you to change how the Scene is displayed (Bounding Box, Wireframe, Solid, Textured).

View Settings

The View Settings section in the properties panel contains addition display options.

Show Overexposed
: Increasing this number to 1 or greater displays a striped overlay to the preview image, showing where it is overexposed. A higher number gives a higher threshold for marking overexposure.

Safe Margin
: Displays an overlay on the preview, marking where title safe region is.

Proxy Render Size
: Draws preview using full resolution or different proxy resolutions. Render resolution is determined in the render settings panel. Using a smaller preview size will increase speed.

Refresh View

Certain operations, like moving an object in 3D View, may not force the Sequencer to call for a refresh of the rendered image (since the movement may not affect the rendered image). If an image or video, used as a strip, is changed by some application outside of Blender, Blender has no real way of being notified from your operating system. To force Blender to re-read in files, and to force a re-render of the 3D View, click the Refresh button to force Blender to update and synchronize all cached images and compute the current frame.

Selecting Strips

The Select Menu helps you select strips in different ways.

Strips to the Left
: Select all strips to the left of the currently selected strip.

Strips to the Right
: Select all strips to the right of the currently selected strip.

Select Surrounding Handles AltCtrl RMB
: Select both handles of the strip, plus the neighboring handles on the immediately adjoining strips. Select with this method to move a strip that is between to others without affecting the selected strip's length.

Left Handle Alt RMB
: Select the left handle of the currently selected strip.

Right Handle Ctrl RMB
: Select the right handle of the currently selected strip.

Linked

Select all strips linked to the currently selected strip

Select All A
: Selects all the strips loaded.

Select Inverse
: Inverts the current selection.

Border Select B
: Begins the Box mode select process. Click and drag a rectangular lasso around a region of strips in your Sequence workspace. When you release the mouse button, the additional strips will be selected.

Moving and Modifying Strips

G Moves the selected strip(s) in time or in channels. Move your mouse horizontally (left/right) to change the strip's position in time. Move vertically (up/down) to change channels.

- To snap while dragging hold Ctrl
- To 'ripple edit' (Make room for strips you drag) hold Alt when placing a strip.

If you have added a strip by mistake or no longer want it, delete it by pressing X or using this menu option.

Duplicate a strip to make an unlinked copy; drag it to a time and channel, and drop it by LMB click.

The Strip Menu contains additional tools for working with strips:

Grab/Move
Grab/Extend from Frame
Cut (hard) at frame
Cut (soft) at frame
Separate Images
Deinterlace Movies

Duplicate Strips
Erase Strips
Set Render Size
Make Meta Strip
UnMeta Strip
Reload Strips
Reassign Inputs
Swap Inputs

Lock Strips
UnLock Strips
Mute Strips
Un-Mute Strips
Mute Deselected Strips
Snap Strips
Swap Strips

Snap to Frame

⇧ ShiftS Position your cursor (vertical green line) to the time you want. Snap to current frame to start a strip exactly at the beginning of the frame. If your Time display is in seconds, you can get to fractional parts of a second by zooming the display; you can get all the way down to an individual frame.

Separate Images to Strips

Y Converts the strip into multiple strips, one strip for each frame. Very useful for slide shows and other cases where you want to bring in a set on non-continuous images.

Editing Strips

- RMB in the middle of the strip selects the **entire** strip; holding it down (or pressing Grab) and then moving the mouse drags a strip around.
- RMB on the left arrow of the strip selects the **start** frame offset for that strip; holding it down (or pressing Grab and then moving the mouse left/right changes the start frame within the strip by the number of frames you move it:
  - If you have a 20-image sequence strip, and drag the left arrow to the right by 10 frames, the strip will start at image 11 (images 1 to 10 will be skipped). Use this to clip off a rollup or useless lead-in.
  - Dragging the left arrow left will create a lead-in (copies) of the first frame for as many frames as you drag it. Use this when you want some frames for transitions to the this clip.
- RMB on the right arrow of the strip selects the **end** frame of the strip; holding it down (or pressing Grab) and then moving the mouse changes the ending frame within the strip:
  - Dragging the right arrow to the left shortens the clip; any original images at the tail are ignored. Use this to quickly clip off a rolldown.
  - Dragging the right arrow right extends the clip. For movies and images sequences, more of the animation is used until exhausted. Extending a clip beyond its end results in Blender making a copy of the last image. Use this for transitions out of this clip.

Multiple selection
You can select several (handles of) strips by ⇧ Shift RMB -clicking: when you'll hit G, everything that's selected will move with your mouse – this means that, for example, you can at the same time move a strip, shorten two others, and extend a forth one.

- STRIP EXTEND. With a number of Image strips selected, pressing E enters EXTEND mode. All selected strip handles to the "mouse side" of the current frame indicator will transform together, allowing you to essentially extend the strips that fall exactly on the current frame marker and having all others adjust to compensate.

While splicing two strips happens just by placing them finish-to-start, cut a strip by pressing K to cut. At the selected frame for the selected strips, K cuts them in two. Use Cut to trim off roll-ups or lead-ins, or roll-downs or extra film shot ("C" was already taken for Change).

Note on the 'cut'
When you 'cut' a strip, you don't really make a cut like it was with the 'old editing' on real film. In fact, you make a copy of the strip: the end of the original one is 'winded' to the cut point, as with the beginning of the new copy.

For example, imagine that you have a strip of **50** frames, and that you want to delete the first ten ones. You have to go to the **11th** frame, and hit K; the cut 'divides' your strip in two parts. You now can select the first small part (frames **1** to **10**), and delete it hitting X.

You might think that you have really erased the frames **1** to **10**, but there are still there, 'winded', as in a film reel, under your frame **11**: you just have deleted one of the two copies of your strip created by the 'cut'. And you can at any time get your 'lost' frames back (just RMB -click on the left arrow of the strip, then G grab it to the left to display the desired number of frames again (or to the right to 'hide' more frames – this is another way to remove frames at the beginning/end of a strip!).

This is at the heart of nearly every editor solution, and that's quite handy!

Action Stops
When extending the start beyond the beginning or end after the ending, keep in mind that only the last image copies, so when viewed, action will stop on that frame. Start your transition (fade, cross) a little early while action is still happening so that the stop action is not that noticeable (unless, of course, you want it to be, like the 80's drama sitcoms).

Change the length of an effect strip by changing the start/end frame of the origin strips.

Copy and Paste

You can copy a clip and paste it using the two header buttons.

Meta Strips

A Meta-Strip is a group of strips. Select all the strips you want to group, and Ctrl-g to group them into one meta. The meta spans from the beginning of the first strip to the end of the last one, and condenses all channels into a single strip, just like doing a mixdown in audio software. Separating (ungrouping) them restores them to their relative positions and channels.

The default blend mode for a meta strip is Replace. There are many cases where this alters the results of the animation so be sure to check the results and adjust the blend mode if necessary.

One convenient use for meta strips is when you want to apply the same effect to multiple strips. For example: scaling a loop. Until blender gets a Loop effect, the only way to loop a clip is to duplicate it several times. If the clip needs any transforms (like scaling or translating an animated watermark or source material in a different aspect ratio) it is much more convenient to apply a single set of transforms to a meta strip built from the repeated duplicates than apply copies of those transforms to each instance in the loop.

It is possible to edit the contents of a meta strip by selecting it and pressing Tab. You can press Tab again to finish editing that strip. Since meta strips can be nested, to pop out one level of meta strip make sure you do not have a meta strip as the active strip when you press Tab.

Sequence Display Modes

By default, the VSE only displays the sequencer. Several options in the header bar allow you change the editor to display the sequence in real time, and in various ways.

The second button will change the editor to display only the preview, and the third button displays both the sequencer and the preview

the VSE workspace can show you different aspects of the composite result, for the current frame:

- Image/Sequence: Colors (what you see)
- Chroma: Color hue and saturation
- Luma: Brightness/contrast
- Histogram: Levels of red, green, and blue

In the Chroma, Luma, and Image modes, a channel selector appears; channel 0 is the result of compositing the strips with their special effects strips. Channel 1 is what the current frame's image from the strip in channel 1 looks like (channel 1 is at the bottom of the heap). The display of these modes is either the composite (channel 0) or the frame from the strip (channels 1 through n).

Zoom the view of any of these workspaces by scrolling your middle mouse wheel.

Image Preview

In the upper window pane of the Sequence screen layout is another VSE window, this one set to Image Preview mode. It shows you what the resulting video will look like when saved. This is the main working mode for adding strips and moving them around, cutting, grouping (making meta) and splicing them through special effects.

Luma Waveform

For the selected channel, brightness, or luminosity, is mapped with this display.

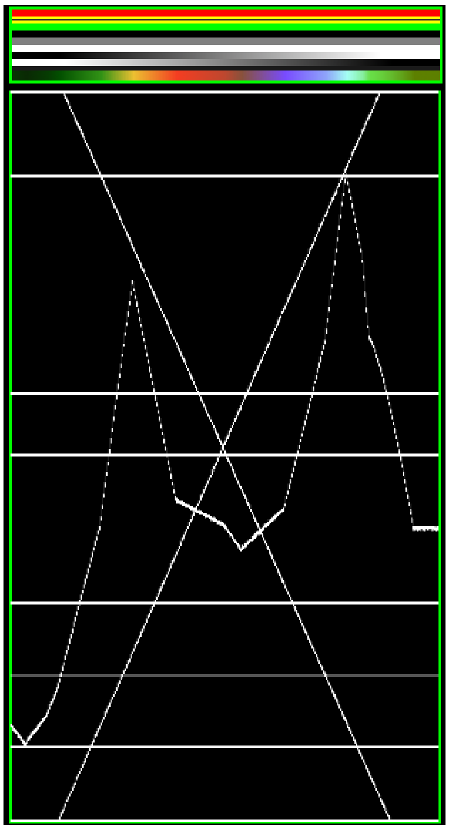
A luma waveform allows you to judge the quality of the luminance distribution across your video signal, you can view a luma-waveform instead of the usual output display on every control monitor.

The display plots for every scanline the luminance value. The lines are all drawn on top of each other. The points get brighter if the lines cross (which is very likely with several hundred scanlines). You will understand the picture most easily if you plug an oscilloscope to the Luma-video-output of your television set. It will basically look the same.

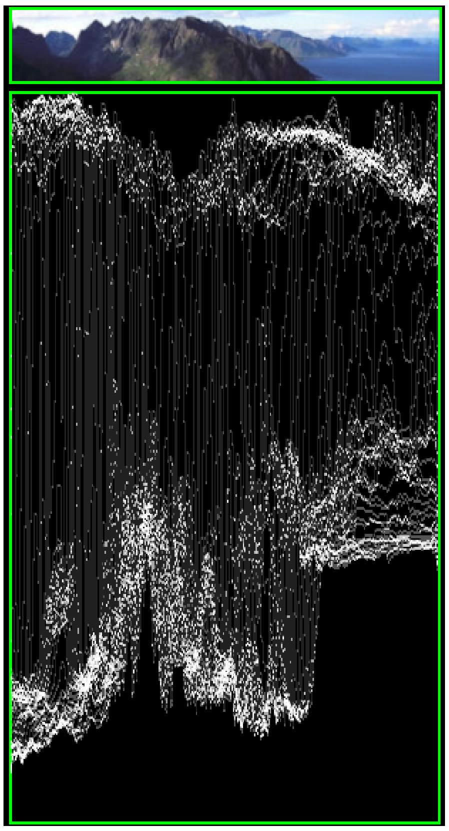
In this mode, the vertical axis represents the luminosity: 0 at the bottom, 1 at the top; the horizontal axis is a mapping from the horizontal axis of the frame. There are as many curves as scanlines in the frame: each one of this curves represents the luminosity of the pixels of one line. Moreover, the color of a pixel in this mode represents the number of pixels from the matching column of the frame sharing the same luminosity – i.e. the number of curves that cross at this point (black/transparent, for no pixel, white/opaque for at least 3 pixels).

This mode is good for:

- If the waveform does not fill the whole picture you might want to play with the "setup" and "gain" master-sliders in the "gamma"-plugin until it fills the whole picture (contrast autostretch).
- With the more advanced gamma-plugin you can decide where you have to desaturate (especially in dark regions).
- You can judge if you want to dump the whole thing since it is completely distorted and clips at the top or the bottom.



'Simple' picture.
The various horizontal lines in the Luma waveform match the uniform-color lines of the picture.
Note that the 'grey 20%' one-pixel width line (inside the yellow strip) is represented in the Luma waveform by a grey line.
The two lines drawing an 'X' are from the two linear tone shades (white→black and black→white).
Finally, the broken line matches the complex tone shade at the bottom of the picture.

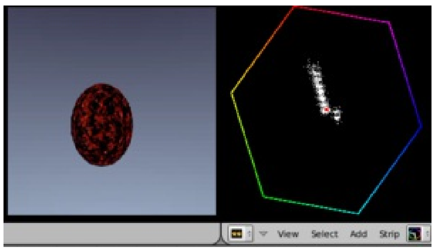


A 'real' picture.
The curves are quite visible.
We found a luma of 80-100% for the sky,
a luma around 40% for the sea,
and a luma of 10-20% for the mountains,
growing around 40% for the sunny part.

Examples of VSE Luma Previews.
Note that the pictures (first green frame, at the top) are only 50px high, to limit the number of curves displayed in the Luma waveform!

Use this display to check for appropriate contrast and luminosity across all frames in the channel. When spots in the film that should have even illumination don't, it looks like a flashbulb went off or an extra light was suddently turned on. This can happen if two strips were rendered or shot under different lighting conditions but are supposed to be contiguous.

Chroma Vectorscope



Example VSE Chroma Preview

Use this mode judge the quality of the color-distribution and saturation, you can also view a U/V scatter-plot.

The picture is converted to YUV-format. The U- and V-values represent the angle of the color. For pixel of the picture, one point is plotted in the display at the U and V-value-position. If several pixels happen to have the same U/V-value the pixel in the plot gets brighter.

To help you understand what color is meant, a hexagram marking the extreme positions (red, magenta, blue, cyan, green, yellow) is drawn and a red cross to mark the origin.

In other words, for the selected channel, this display shows the color space of the image inside a hexagon. Each point of the hexagon is a primary color: red, magenta, blue, cyan, green, and yellow. Black is at the center, and overall saturation is scaled as dots closer to the outside. The example to the right shows that the image has a lot of red (50% saturation) and small amount of blue, with no green.

Always: remember to activate an additional control monitor of the end result. Color calibration is a matter of taste and depends on what you want.

Use this display to check for too much color saturation. While over-saturated images look great for op-art and computer displays, they stink when shown on the big screen TV. Use the Alt-Animation key to scrub the video; this display will update with a new/revised map for each frame. Just like watching the Image preview to see what it looks like, watch the Chroma Vectorscope to watch for color use.

This mode is good for:

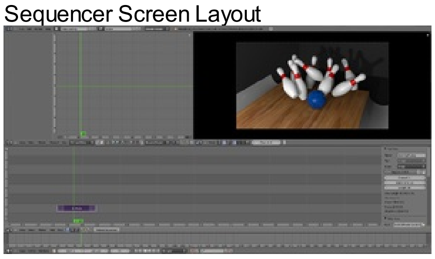
- If you picture looks very moody or desaturated you might want to take a look at the U/V-plot. You will most likely see all pixels building a crowd at the origin. If you add saturation using the "gamma"-plugin you can see in the U/V-plot if you distort the color.
- If you do color-matching on a by hand basis you can match the angle you see of different channels monitors.

Histogram

This mode displays a graph showing the distribution of color information in the pixels of the currently displayed image. The X-axis represents values of pixel, from 0 to 1 (or 0 to 255), while the Y-axis represents the number of pixels in that tonal range. A predominantly dark image would have most of its information toward the left side of the graph.

Use this mode to balance out the tonal range in an image. A well balanced image should a nice smooth distribution of color values.

Sequencer Screen Layout



Default Video Editing screen lay out

Sequencer Effects

Blender offers 16 built effects that are built into Blender, and are therefore universal. Some operate on two strips; some on one, and some create a new strip. Each effect enhances your content in some way or allows professional-quality transitions.

Add

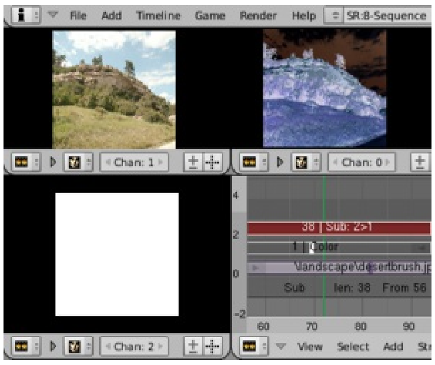
Can you hear the thunder?

The Add effect adds two colors together. Red and Cyan (Green and Blue) make White. Red and Blue make "Magenta" (i.e. Purple!). Red and Green make Yellow.

The Add Effect adds the colors of two strips together, Use this effect with a base image strip, and a modifier strip. The modifier strip is either a solid color or a black-and-whte mask, or another image entirely. The example to the right shows what happens when you add gray to an image, and animate the effect over time. The image gets bright because we are adding gray (R:.5, G:.5, B:.5) to say, a blue color (R.1, G:.1, B:.5) resulting in (R:.6, G:.6, B:1.0) which retains the original hue (relationship between the colors) but is much brighter (has a higher value). When applied to the whole image like this, the whole image seems to flash.

You can use this effect to increase the brightness of an image, or if you use a BW mask, selectively increase the brightness of certain areas of the image. The Mix node, in Add mode, does exactly the same thing as the Add sfx strip here, and is controlled the same way by feeding the Factor input.

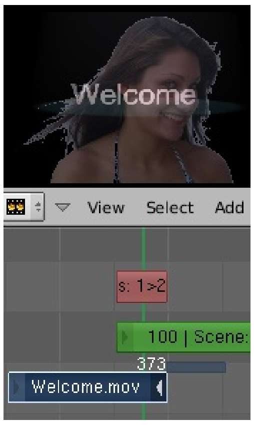
Subtract Effect



Subtract Effect

This effect takes away one strip's color from the second. Make a negative of an image using this effect, or switch the order of the strips and just darken the strip. Subtracting a hue of blue from a white image will make it yellow, since red and green make yellow.

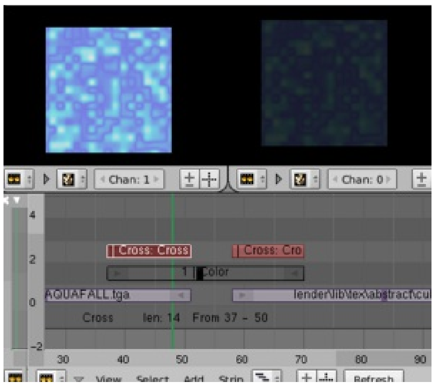
Cross and Gamma Cross



This effect fades from one strip to another, based on how many frames the two strips overlap. This is a very useful strip that blends the whole image from one to the other.

Gamma Cross uses color correction in doing the fade, resulting in a smooth transition that is easier on the eye.

Fade to Black

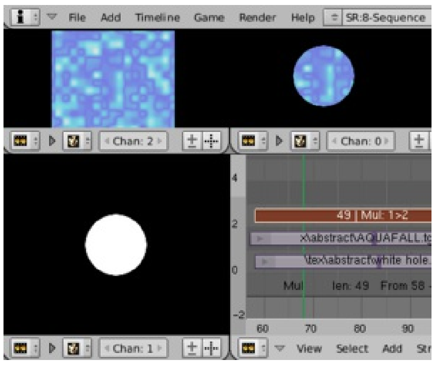


Cross-Fade between Black

Many scenes fade to black, and then fade in from black, rather than directly from one to the other.

The strip setup to do this is shown to the right. The two strips are on Channel 1, and you Add->Color Generator strip to Channel 2, straddling the two main strips. Change the color to black, and add two Cross Effects; the first from Channel 1 to Channel 2 (black), and the second from Channel 2 to Channel 1. The first strip will fade to black, and then the second will fade in from black. Of course, you can use any transition color you want. Black is a relaxing intermediary; red is alarming. Use the dominant color in the second strip to introduce the second strip.

Multiply



Multiply Effect.

The Multiply effect multiplies two colours. Blender uses values between **0.0** and **1.0** for the colours, he doesn't have to normalise this operation, the multiplication of two terms between **0.0** and **1.0** always gives a result between **0.0** and **1.0** (with the 'traditional' representation with three bytes – like RGB(**124**, **255**, **56**) –, the multiplications give far too high results – like RGB(**7316**, **46410**, **1848**) –, that have to be 'brought back', normalised – just by dividing them by **256**! – to 'go back' to range of **0** to **255**…).

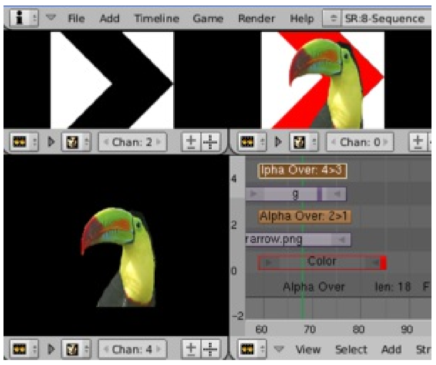
This effect has two main usages:

With a mask
: A mask is a B&W picture witch, after multiplication with a 'normal' image, only show this one in the white areas of the mask (everything else is black). The opening title sequence to James Bond movies, where the camera is looking down the barrel of a gun at James, is a good example of this effect.

With uniform colors
: Multiplying a color with a 'normal' image allows you to soften some hues of this one (and so – symmetrically – to enhance the others). For example, if you have a brown pixel RGB(**0.50**, **0.29**, **0.05**), and you multiply it with a cyan filter (uniform color RGB(**0.0**, **1.0**, **1.0**), you'll get a color RGB(**0.0**, **0.29**, **0.5**). Visually, the result is to kill the reds and bring up (by 'symmetry' – the real values remain unchanged!) the blues an greens. Physically, it is the same effect as shining a cyan light onto a chocolate bar. Emotionally, vegetation becomes more lush, water becomes more Caribbean and inviting, skies become friendlier.

Note
This effect reduces the global luminosity of the picture (the result will always be smaller than the smallest operand). If one of the image is all white, the result is the other picture; if one of the image is all black, the result is all black!

Alpha Over, Under, and Over Drop



AlphaOver Effect

Using the alpha (transparency channel), this effect composites a result based on transparent areas of the dominant image. If you use a Scene strip, the areas of the image where there isn't anything solid are transparent; they have an alpha value of 0. If you use a movie strip, that movie has an alpha value of 1 (completely opaque).

So, you can use the Alpha Over/Alpha Under effect to composite the CGI Scene on top of your movie. The result is your model doing whatever as if it was part of the movie. The Factor curve controls how much the foreground is mixed over the background, fading in the foreground on top of the background. The colors of transparent foreground image areas is ignored and does not change the color of the background.

Select two strips (⇧ Shift RMB 🖱):

- With Alpha Over, the strips are layered up in the order selected; the first strip selected is the background, and the second one goes *over* the first one selected. The Factor controls *the transparency of the foreground*, i.e. a Fac of **0.0** will only show the background, and a Fac of **1.0** will completely override the background with the foreground (except in the transparent areas of this one, of course!)
- With Alpha Under, this is the contrary: the first strip selected is the foreground, and the second one, the background. Moreover, the Factor controls *the transparency of the background*, i.e. a Fac of **0.0** will only show the foreground (the background is completely transparent), and a Fac of **1.0** will give the same results as with Alpha Over.

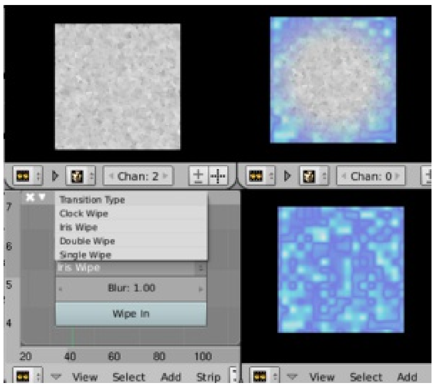
- Alpha Over Drop is between the two others: as with Alpha Under, the first strip selected will be the foreground, but as with Alpha Over, the Factor controls the transparency of this foreground.

The example shows layering of AlphaOver effects. The very bottom channel is red, and an arrow is on top of that. Those two are AlphaOver to Channel 3. My favorite toucan is Channel 4, and Channel 5 alphaovers the toucan on top of the composited red arrow. The last effect added is tied to Channel 0 which will be rendered.

By clicking the PreMult Alpha button in the properties panel of the foreground strip, the Alpha values of the two strips are not multiplied or added together. Use this effect when adding a foreground strip that has a variable alpha channel (some opaque areas, some transparent, some in between) over a strip that has a flat opaque (Alpha=1.0 or greater) channel. If you notice a glow around your foreground objects, or strange transparent areas of your foreground object when using AlphaOver, enable PreMultiply. The AlphaOver Drop effect is much like the Cross, but puts preference to the top or second image, giving more of a gradual overlay effect than a blend like the Cross does. Of course, all of the Alpha effects respect the alpha (transparency) channel, whereas Cross does not.

The degree of Alpha applied, and thus color mixing, can be controlled by an F-curve. Creating a Sine wave could have the effect of the foreground fading in and out.

Wipe



VSE Wipe Built-in Effect

Wipe transitions from one strip to another. This very flexible effect has four transition types:

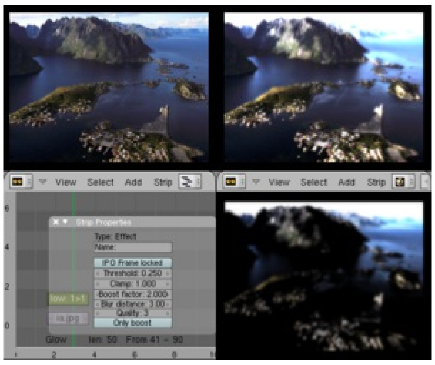
- Clock: like the hands of an analog clock, it sweeps clockwise or (if Wipe In is enabled) counterclockwise from the 9:00 position. As it sweeps, it reveals the next strip.
- Iris: like the iris of a camera or eye, it reveals the next strip through an expanding (or contracting) circle. You can blur the transition, so it looks like ink bleeding through a paper.
- Double Wipe: Starts in the middle and wipes outward, revealing the next strip. It can also Wipe In, which means it starts at the outside and works its way toward the middle. You can angle and blur the wipe direction as well.

- Single Wipe: Reveals the next strip by uncovering it. Controls include an angle control so you can start at a corner or side, and blur the transition.

The wipe will have no effect if created from a single strip instead of two strips. The duration of the wipe is the intersection of the two source strips and can not be adjusted. To adjust the start and end of the wipe you must adjust the temporal bounds of the source strips in a way that alters their intersection.

Note: some older plugins contain similar functionality.

Glow



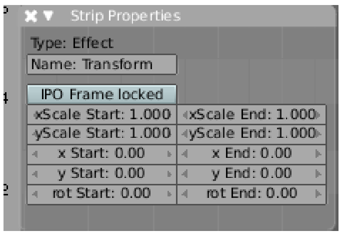
Type: Effect
Name:
IPO Frame locked
Threshold: 0.250
Clamp: 1.000
Boost factor: 2.000
Blur distance: 3.00
Quality: 3
Only boost
Glow len: 50 From 41 – 90

Example of a Glow effect applied to a picture.
Top left: base picture (Lofoten Islands, Norway – source: wikipedia.fr);
Top right: result of the effect;
Bottom left: effect settings;
Bottom right: result with the Only boost button activated.

This effect makes parts of an image glow brighter by working on the luminance channel of an image. The Glow is the superposition of the base image and a modified version, where some areas (brighter than the Threshold:) are blurred. With the Glow strip properties, you control this Threshold:, the maximum luminosity that can be added (Clamp:), a Boost factor: for it, the size of the blur (Blur distance:), and its Quality:. The Only boost button allows you to only show/use the 'modified' version of the image, without the base one. To "animate" the glow effect, mix it with the base image using the Gamma Cross effect, crossing from the base image to the glowing one.

Transform

(Note: Transform does not work in Blender 2.49) Transform is a swiss-army knife of image manipulation. It scales, shifts, and rotates the images within a strip. The example to the right shows what can be done with a single image. To make a smooth transition to the final effect, enable the Frame locked button and define a curve in the Ipo Window (Sequence mode).



Type: Effect
Name: Transform
IPO Frame locked

xScale Start: 1.000	xScale End: 1.000
yScale Start: 1.000	yScale End: 1.000
x Start: 0.00	x End: 0.00
y Start: 0.00	y End: 0.00
rot Start: 0.00	rot End: 0.00

With the Transform strip selected, uses the properties panel to adjust the settings of this effect:

(x,y)Scale (Start,End):
: To adjust the scale (size). xScale Start defines the start width, xScale End the end width, yScale Start the start height, and yScale End the end height. The values higher than **1.0** will scale up the picture, while values lower than **1.0** will scale it down.

(x,y) (Start,End):
: To adjust the position (shifting). x Start defines the horizontal start position, x End, the end one; positive values shift the image to the right, negative values, to the left. y Start defines the vertical start position, y End, the end one; positive values shift the picture to the top, negative values, to the bottom.

rot (Start,End):
: The rotation is in degrees (**360** for a full turn) and is counter-clockwise. To make an image spin clockwise, make the end value lower than the start one (e.g. start it at 360 and go down from there).

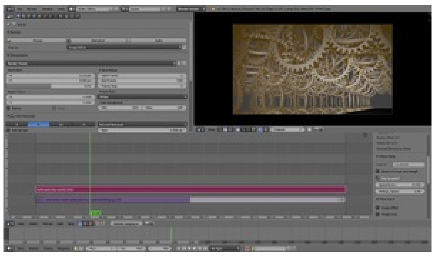
Color

This effect works by itself to create a color strip. By default, when it is created, it is 50 frames long, but you can extend it by grabbing and moving one of the ends. Click on the color swatch in the Effect panel under Sequencer buttons, which is under the Scene (F10) tab, to pick a different color (by default, it is gray). Use this strip crossed with your main movie to provide a fade-in or fade-out.

Speed Control

Speed Control time-warps the strip, making it play faster or slower than it normally would. A Global Speed less than 1.0 makes the strip play slower; greater than 1.0 makes it play faster. Playing faster means that some frames are skipped, and the strip will run out of frames before the end frame. When the strip runs out of frames to display, it will just keep repeating the last one; action will appear to freeze. To avoid this, position the next strip under the original at a point where you want motion to continue.

Creating a Slow-Motion Effect


50% Slow motion using Speed Control

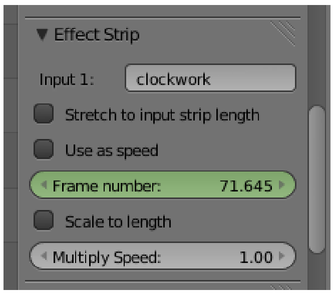
Suppose you want to ssssooow your strip dowwwwwwn. You need to affect the speed of the video clip without affecting the overall frame rate. Select the clip and Add->Effect->Speed Control effect strip. Click to drop it and press N to get the Properties. Uncheck the *Stretch to input strip length* option in the Effect Strip section. Set the Speed factor to be the factor by which you want to adjust the speed. To cut the displayed speed by 50%, enter 0.50. Now, a 275-frame clip will play at half speed, and thus display only the first 137 frames.

If you want the remaining frames to show in slo-mo after the first set is displayed, double the Length of the source strip (since effects strip bounds are controlled by their source strips). If you're using a speed factor other than 0.5 then use the formula

```
new_length = true_length / Speed_factor
```

That's it! Set your render to animate (in this example) all 550 frames.

Keyframing the Speed Control


▼ Effect Strip

Input 1: clockwork

☐ Stretch to input strip length

☐ Use as speed

Frame number: 71.645

☐ Scale to length

Multiply Speed: 1.00

To get even finer control over your clip timing, you can use curves! While it is possible to keyframe the Speed factor, usually you want to keyframe the Frame number directly.

Uncheck *Stretch to input strip length* and uncheck *Use as speed*. You now have a Frame number field which you can keyframe. If you want the strip to animate **at all** you will have to insert some keyframes, otherwise it will look like a still. In most cases you will want to use the Graph editor view to set the curve interpolation to Linear since the default Bezier will rarely be what you want.

If you do choose to keyframe the Speed factor instead, remember to click the Refresh Sequencer button in the header of the Video Sequence Editor's strip view or your changes will not take effect.

Changing Video Frame Rates

You can use the speed control to change the frames per second (fps), or framerate, of a video. If you are rendering your video to a sequence set, you can effectively increase or decrease the number of individual image files created, by using a Global Speed value less than or greater than one, respectively. For example, if you captured a five-minute video at 30 fps and wanted to transfer that to film, which runs at 24 fps, you would enter a Global Speed of 30/24, or 1.25 (and Enable Frame Blending to give that film blur feel). Instead of producing 5*60*30=9000 frames, Blender would produce 9000/1.25=7200=5*60*24 frames. In this case, you set a Sta:1 and End:7200, set your Format output to Jpeg, 30fps, and image files 0001.jpg through 7200.jpg would be rendered out, but those images 'cover' the entire 9000 frames. The image file 7200.jpg is the same a frame 9000. When you read those images back into your film .blend at 24 fps, the strip will last exactly 5 minutes.

Multicam Selector

Ever wanted to do multicam editing with Blender? Now you can and it is mindbogglingly easy:

- Add your input strips on channels say 1 to 4 (you can use as many you like, interface get's a little bit clumky if you have more than 10, see below).
- Sync the strips up. There is no automatic sync feature in Blender, but you can open two viewer windows, choose one camera as the master channel and sync the other against them just by looking at the movement of legs or light flashes (depending of the show, you want to edit). We might add automatic sync feature based on global brightness of the video frames in the future. (Syncing based on the audio tracks, like most commercial applications do, isn't very clever, since the speed of sound is only around 340 metres per second and if you have one of you camera 30 meters away, which isn't uncommon, you are already 2-3 frames off. Which *is* noticeable...)
- Build small resolution proxies (25%) on all your input video strips.
- Use meta strips, so that every input camera fits in exactly one channel.
- Add a viewer window for every input channel and put it into 25% proxy display mode (I suggest to line them up on the left side on top of each other, but just do, whatever pleases your personal habits)
- Add a large viewer window for the final output and let it run on full resolution.
- Add a multicam selector effect strip *above* all the channel tracks
- Enlarge it, so that it covers the whole running time of your show (just change it's length or drag the right handle, the former is probably easier, since you can just type in a very large number and you are done)
- Cross you fingers :) (that's important :))
- Select the multicam strip, if you take a look at the strip options (N-key), you will notice, that multicam is a rather simple effect strip: it just takes a selected channel as it's input. That's all. The magic comes with the convenient keyboard layout: when you select multicam, the keys 1-0 are mapped to a python handler, that does a cut on the multicam and changes it's input.
- So: you select the multicam strip, you start playback and hit the keys 1-4 while watching your show.
- You'll end up with a small multicam selector strip for every cut.

In reality, it boils down to: watch a few seconds to see, what's coming, watch it again and do a rough cut using the number keys, do some fine tuning by selecting the outer handles of two neighboring multicam for A/B rolling.

Adjustment Layer

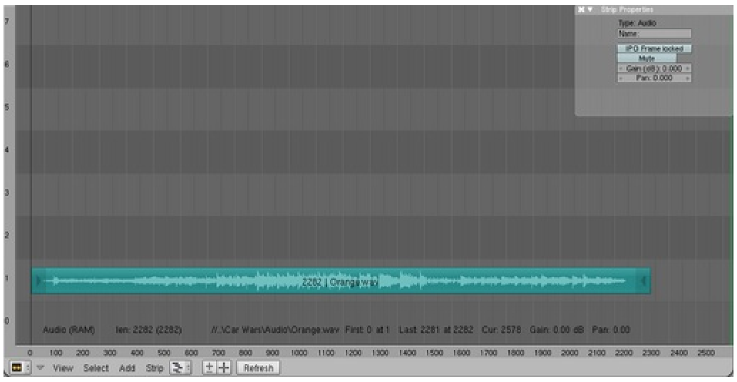
The adjustment layer strip works like a regular input file strip except for the fact, that it considers all strips below it as it's input.

Real world use cases: you want to add some last finishing color correction on top of parts of your final sequencer timeline without messing with metastrips around. Just add an adjustment layer on top and activate the color balance.

Or: you can stack a primary color correction and several secondary color correction on top of each other (probably using the new mask input for area selection).

Sound Editing

Blender contains a multi-track Audio sequencing toolbox. You can add WAV, Mp3 files from your hard disk as a file, or as encoded within a movie, and mix them using an F-Curve as a volume control.



Audio (RAM) len: 2282 (2282) //..\Car Wars\Audio\Orange.wav First: 0 at 1 Last: 2281 at 2282 Cur: 2578 Gain: 0.00 dB Pan: 0.00

A sound strip in the sequence editor.

Options

Audio-RAM loads a file into memory and plays it from there. You can only load stand-alone WAV files. Audio-HD plays the sound back from the hard disk and thus does not take up memory. With Audio HD, you can load stand-alone WAV files, but also audio tracks from movies.

For either, a green audio strip will be created. With Audio RAM, a waveform is created that shows you the waveform inside the green strip, scaled to the height of the green strip. Since Audio-RAM files are read into memory, changing the audio file will not affect playback, and you will have to re-open the file so that Blender re-reads the file.

Hiss, Crackle and Pop
Some audiophile users report that Hiss is introduced sometimes if Audio RAM is used. There must be some decoding or sampling going on, that does not occur when Audio HD is used, that introduces some playback noise. If you hear pops and crackles, usually that is a sign that your hardware cannot keep up in real-time playback. They will not be present in your final rendered animation output (but they may show up in Game mode). Also, static hiss seems to occur whenever two or more audio strips are overlapping in the timeline…

Audio Mixing in the VSE

You can have as many Audio strips as you wish and the result will be the mixing of all of them. You can give each strip its own name and Gain (in dB) via the N menu. This also let you set a strip to mute or 'Pan' it; -1 is hard left, +1 is hard right, with percentages in-between.

Overlapping strips are automatically mixed down during ANIM processing. For example, you can have the announcer on channel 5, background music on channel 6, and foley sound effects on channel 7.

Working with Audio Tracks

An audio track (strip) is just like any other strip in the VSE. You can grab and move it, adjust its starting offset using RMB over the arrow end handles, and K cut it into pieces. A useful example is cutting out the "um's" and dead voice time.

Animating Audio Track Properties

You want to set a value somewhere between 0.0 and 1.0, and the volume becomes that percent; 0.6 is 60%. You can add a gain to the volume through the strip properties (N). You can make a curve by having multiple points, to vary the volume over its length. Press ⇆ Tab to edit the curve, just like any old bezier F-curve.

In the Y direction, 1.0 is full volume, 0.0 is completely silent. Only the FFMPEG-output system is currently able to mix audio and video into one output stream. Use Ctrl LMB to add control points, and ⇆ Tab to edit a curve.

Animating an audio strip affects the volume of the strip in the resulting composite. Use animation on an audio strip to fade in/out background music or to adjust volume levels. Layered/crossed audio strips are added together; the lower channel does not override and cut out higher channels. This makes Blender an audio mixer. By adding audio tracks and using the curves to adjust each tracks' sound level, you have an automated dynamic multi-track audio mixer!

Output

The output is therefore a video file if the ANIMATION button in the Render Panel of the Scene Context/Render Sub-context is used as described before. An audio file may be created via the MIXDOWN button in the Sequencer button of the Scene Context, Sound Sub-context. This WAV file contains the full audio sequence and is created in the same directory of the video file and with the same name but with a .WAV extension. You can mix Video and Audio later on with an external program or by adding it to, for example, an image sequence strip as described above.

The advantage of using Blender's sequence editor lies in the easier synchronization attainable by sequencing frames and sound in the same application.

To enable audio synchronisation after importing an audio track, select the Scene button (F10) in the buttons window then choose the Sound Block Button (small blue sine wave). In here you'll see the Sync and Scrub tools.

- Sync lets Blender drop image frames to keep up with realtime audio when you play an animation in the 3D window. This gives you a rough overview of the timing of your animation.
- Scrub allows you to drag your frame-marker or change frames in any window and it will play a clip of audio for that point in time.

Draging the frame-marker over a range of frames in the Action Editor will allow you to hear roughly where specific sounds occur so that you can key poses or shapes on this frame.

Extending Blender

Unlike many programs you may be familiar with, Blender is not monolithic and static. You can extend its functionality with Python scripting without having to modify the source and recompile.

Addons

Addons are scripts you can enable to gain extra functionality within Blender, they can be enabled from the user preferences.

Outside of the Blender executable, there are literally hundreds of addons written by many people:

- Officially supported addons are bundled with Blender.
- Other **Testing** addons are included in development builds but not official releases, many of them work reliably and are very useful but are not ensured to be stable for release.

An Overview of all addons is available in this wiki in the Scripts Catalog and in the Extensions tracker.

Scripts

Apart from addons there are also scripts you can use to extend Blenders functionality:

- Modules: Utility libraries for import into other scripts.
- Presets: Settings for Blender's tools and key configurations.
- Startup: These files are imported when starting Blender. They define most of Blender's UI, as well as some additional core operators.
- Custom scripts: In contrast to addons they are typically intended for one-time execution via the text editor

Saving your own scripts

File location

All scripts are loaded from the `scripts` folder of the local, system and user paths.

You can setup an addittional search path for scripts in User preferences (User Preferences → File Paths).

Installation

Addons are conveniently installed through Blender in the User Preferences → Addons window. Click the Install from File... button and select the `.py` or `.zip` file.

To manually install scripts or addons place them in the `addons`, `modules`, `presets` or `startup` directory according to their type. See the description above.

You can also run scripts by loading them in the text editor window.

Blender 2.6, Python Manual

Introduction

Welcome to the Blender 2.6 Python Manual.

Python www.Python.org is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. Python scripts are a powerful and versatile way to extend Blender functionality. Most areas of Blender can be scripted, including Animation, Rendering, Import and Export, Object Creation and the scripting of repetitive tasks.

To interact with Blender, scripts can make use of the tightly integrated API (Application Programming Interface).

General information

Links that are useful while writing scripts.

- Blender Python API
  - Official API documentation. Use this for referencing while writing scripts.
- API introduction
  - A short introduction to get you started with the API. Contains examples.
- CookBook
  - A section of handy code snippets (yet to be written)
- FAQ
  - Frequently asked questions and their answers

Links that deal with distributing your scripts.

- Sharing scripts
  - Information on how to share your scripts and get them included in the official Blender distribution.
- Creating Add-Ons
  - As of Blender 2.5 Alpha1, this is the new way to spread scripts for Blender.
- Extensions project
  - Project to maintain a central repository of extensions to Blender.

Getting Started - Wiki tutorials

The following pages are located on this wiki and take you from the basics to the more advanced concepts of Python scripting for Blender.

- Hello World
- Console
- Text editor
- Geometry
- Properties, ID-Properties and their differences

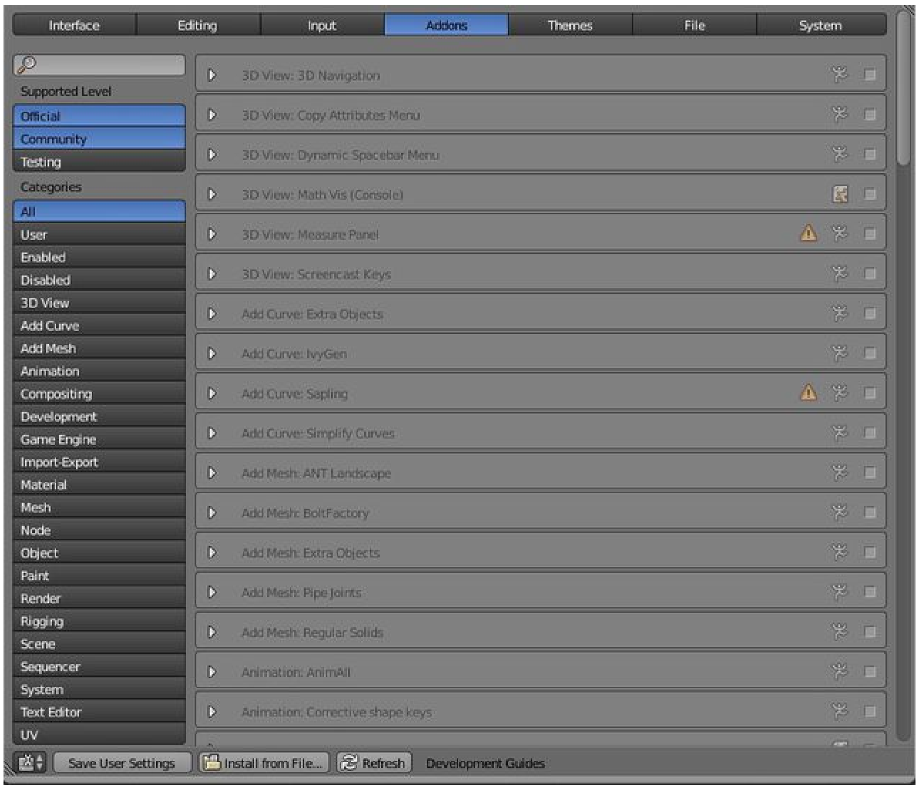
Getting Started - External links

The following pages are not located on this wiki, but contain a lot of good information to start learning how to write scripts for Blender.

- Introductory tutorial by Satish Goda
  - Takes you from the beginning and teaches how to do basic API manipulations.
- Ira Krakow's video tutorials
  - First video in a series of video tutorials.
- Quickstart guide
  - A quickstart guide for people who already have some familiarity with Python and Blender.
- Examples thread
  - A forum thread containing many short working script examples.
- Introduction to Python
  - A one hour video tutorial introducing Python and the Blender API.

Add-Ons

Add-On is the general term for any optional script that extends Blender's functionality. They are found in the Add-Ons tab of the User Preferences window. This tab allows to install, enable and disable Add-Ons. Blender comes with some useful Add-Ons already, but you can also add your own, or any interesting ones you find on the web. The Scripts Catalog provides an index of Add-Ons that are included with Blender as well as listing a number of external Add-Ons.



Installation of an Add-On

For a script to show up in the Add-Ons tab it will first have to be installed. For this you can use the Install Add-On button in the header of the Add-Ons window. Simply click the button and locate the script you wish to install. Once installed, the script will show up in the panel.
Alternatively you can manually install an Add-On. An Add-On is considered installed when it is located in the `../scripts/addons` folder (where .. is the path to your Blender configuration folder). Simply moving the Add-On into that folder is enough.

Addons can be python scripts **.py** or **.zip** files (containing **.py** scripts).

File locations

- Windows 7 - `C:\Users\%username%\AppData\Roaming\Blender Foundation\Blender\2.6x\scripts\addons`
- Windows XP - `C:\Documents and Settings\%username%\Application Data\Blender Foundation\Blender\2.6x\scripts\addons`
- Linux - `/home/$user/.config/blender/$version/scripts/addons`

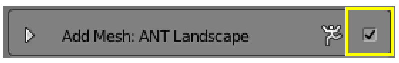
Note that the `AppData` folder in Windows 7 and the `.config` folder in Linux is hidden. The location may also be different depending on your choices for setting up your operating system and Blender.

You can also create a personnal folder containing new addons and configure your files path in the File panel of the User Preferences. To create a personnal script folder:

1. Create an empty folder (i.e. 'script_addon_2-6x')
2. Add one folder named 'addons'. It has to named like this for Blender to recognize it.
3. Put your new addons in this 'addons' folder.
4. open the File panel of the User Preferences.
5. Fill the Scripts entry with the path to your script folder (i.e. 'script_addon_2-6x').

For information on the location of blender directories
see: Configuration & Data Paths

Enabling and Disabling



Enabling an Add-On

Once an Add-On has been installed, it has to be enabled before it can be used. Simply place a check mark on the Enable Add-On box of the Add-On you wish to activate and you're done. The extra functionality of the Add-on is now integrated into Blender and can be used.

To disable the functionality again, uncheck the box. To get more information on a certain Add-on you can press the arrow at the left of the entry and any additional information that is available will be shown. If the Add-On does not activate when enabled, check the Console window for any errors that may have occurred when loading.

Saving Add-On Preferences

If you want an Add-On to be enabled everytime you start Blender, you will need to save your User Preferences.

Development guidelines

If you are a script developer, you may be interested in the Add-Ons development guidelines.

Why another Python tutorial?

This page exists for two reasons.

1. To introduce Programming using Python 3.x quickly and efficiently.
2. And most importantly teach inside of Blender's Console, so you can learn in context.
3. See External links below

What is Programming?

Programming in simple terms is nothing more than manipulating data. Operations on the data either modifies it, or create new data.

The simplest data is Numbers. Operations on Numbers are addition, Subtraction, multiplication etc., Its the simplest type of data imaginable.

But for solving real world problems, we need have compound data, that is built from simpler data like numbers. A good example is Vector data type, that is built from 3 numbers.

During the course of this tutorial, we will take a look at what data types that Python language provides and how you can create your own custom data for your programs and also write operations that work with that data.

What is Python?

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax.

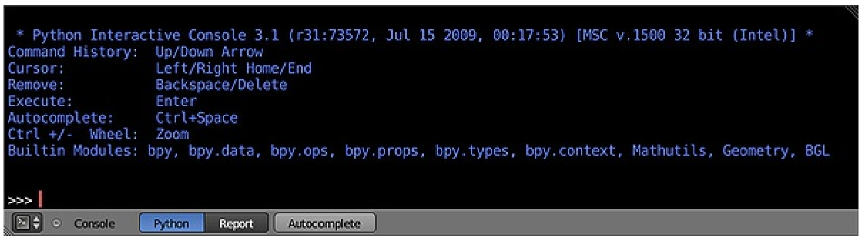
Learning Python is also very easy, even if you have never programmed before.

Python Interpreter

All the exercises in this tutorial will be using the built-in Console window type in Blender 2.6, which has a Python 3.2 interpreter embedded in it.

Following is a video that shows how you can switch to the interpreter.

[video link]



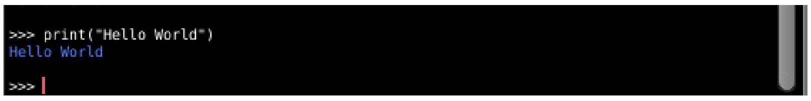
```
 * Python Interactive Console 3.1 (r31:73572, Jul 15 2009, 00:17:53) [MSC v.1500 32 bit (Intel)] *
Command History:  Up/Down Arrow
Cursor:           Left/Right Home/End
Remove:           Backspace/Delete
Execute:          Enter
Autocomplete:     Ctrl+Space
Ctrl +/-  Wheel:  Zoom
Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, Mathutils, Geometry, BGL

>>>
```

You can start typing Python commands, expressions and statements at the interpreter prompt **>>>**

Hello World

Let's get started with the classical "Hello World" program.

Type the following print statement at the interpreter prompt and press ↵ Enter key.



```
>>> print("Hello World")
Hello World

>>>
```

Let's break down the above statement.

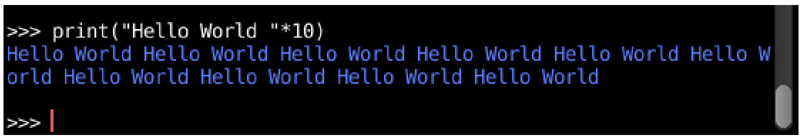
1. *"Hello World"* is a string literal in Python.
   1. A string is a sequence of characters (numbers, alphabets, special characters)
2. *print()* is a built-in function in Python to print output.
3. *print("Hello World")* outputs *Hello World* to the console.

Exercise
Type the following commands and check the output

```
print('"Hello World"')
print("'Hello \n World'")
```

In Python, a string literal can be multiplied by a *number*. By doing so we are repeating the string by the count specified by *number*

- number * string literal
- string literal * number
- ***** is the multiplication operator in Python



```
>>> print("Hello World "*10)
Hello World Hello World Hello World Hello World Hello World Hello W
orld Hello World Hello World Hello World Hello World

>>> 
```

Note
Check out all the above examples in one place

External links

Webpages

- http://www.sthurlow.com/python/

Video Tutorials

- If you want to learn Python programming in general, have a look at this tutorials

[video link]

The Console Editor Type

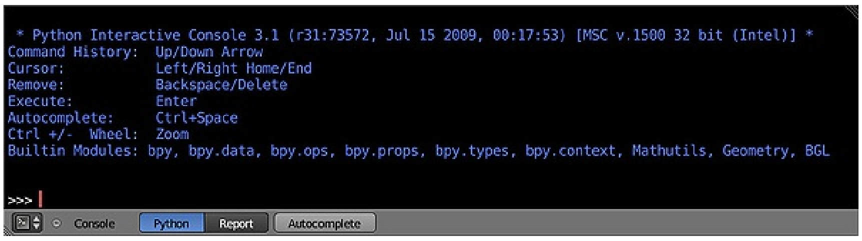
The interactive console in Blender 2.5 has been improved. Auto Complete, Python Reporting & more features have been added. Testing one-liners in the console is a good way to learn the Python API.

Accessing Built-in Python Console

Launching the Console using mouse.

[video link]

By pressing ⇧ ShiftF4 in any Blender Editor Type (3D View, Timeline etc.,) you can change it to a Console Editor.



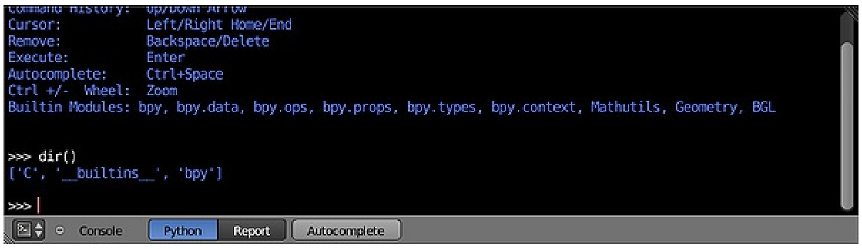
```
 * Python Interactive Console 3.1 (r31:73572, Jul 15 2009, 00:17:53) [MSC v.1500 32 bit (Intel)] *
Command History:  Up/Down Arrow
Cursor:           Left/Right Home/End
Remove:           Backspace/Delete
Execute:          Enter
Autocomplete:     Ctrl+Space
Ctrl +/-  Wheel:  Zoom
Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, Mathutils, Geometry, BGL

>>>
```

From the screen shot above, you will notice that apart from the usual hot keys that are used to navigate, by pressing CtrlSpace you can enable Auto-complete feature.

Since Blender 2.5 uses Python 3.x, the interpreter is loaded and is ready to accept commands at the prompt **>>>**

First look at the Console Environment

To check what is loaded into the interpreter environment, type dir() at the prompt and execute it.



```
Command History:  Up/Down Arrow
Cursor:           Left/Right Home/End
Remove:           Backspace/Delete
Execute:          Enter
Autocomplete:     Ctrl+Space
Ctrl +/-  Wheel:  Zoom
Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, Mathutils, Geometry, BGL

>>> dir()
['C', '__builtins__', 'bpy']

>>>
```

Following is a quick overview of the output

'C'
: Quick access to bpy.context

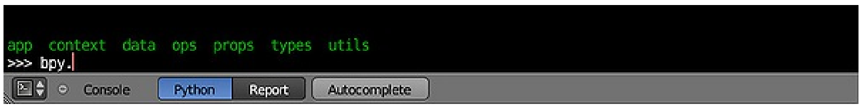
'D'
: Quick access to bpy.data

'__builtins__'
: Python Built-ins (Classes, functions, variables)

'bpy'
: Top level Blender Python API module.

Auto Completion at work

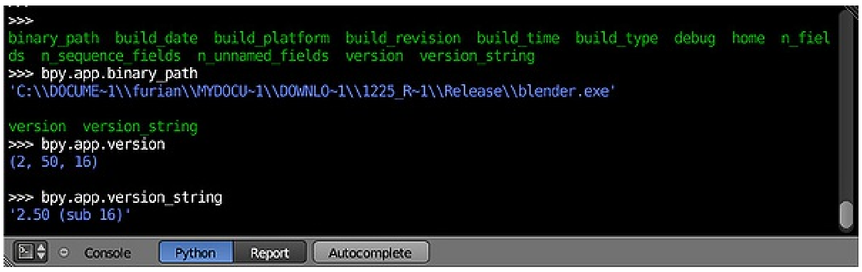
Now, type bpy. and then press CtrlSpace and you will see the Console auto-complete feature in action.



```
app  context  data  ops  props  types  utils
>>> bpy.
```

You will notice that a list of sub-modules inside of bpy appear. These modules encapsulate all that we can do with Blender Python API and are very powerful tools.

Lets list all the contents of bpy.app module.



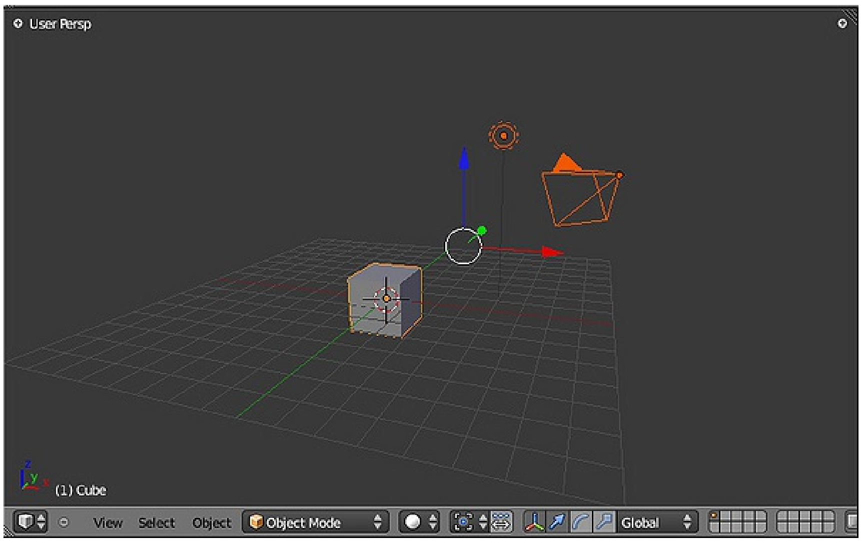
```
>>>
binary_path  build_date  build_platform  build_revision  build_time  build_type  debug  home  n_fiel
ds  n_sequence_fields  n_unnamed_fields  version  version_string
>>> bpy.app.binary_path
'C:\\DOCUME~1\\furian\\MYDOCU~1\\DOWNLO~1\\1225_R~1\\Release\\blender.exe'

version  version_string
>>> bpy.app.version
(2, 50, 16)

>>> bpy.app.version_string
'2.50 (sub 16)'
```

Notice the green output above the prompt where you enabled auto-completion. What you see is the result of auto completion listing. In the above listing all are module attribute names, but if you see any name end with '(', then that is a function.

We will make use of this a lot to help our learning the API faster. Now that you got a hang of this, lets proceed to investigate some of modules in bpy.

Before tinkering with the modules..

If you look at the 3D Viewport in the default Blender scene, you will notice 3 objects: Cube, Lamp and Camera.



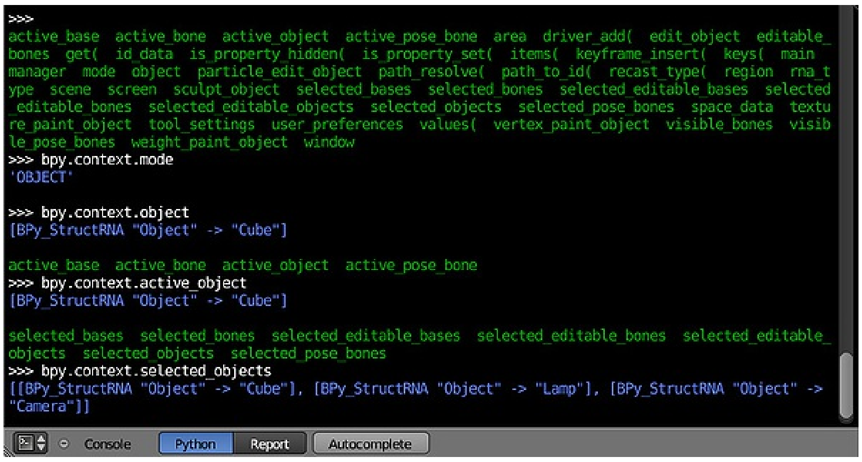
- All objects exist in a context and there can be various modes under which they are operated upon.
- At any instance, only one object is active and there can be more than one selected objects.
- All objects are data in the Blender file.
- There are operators/functions that create and modify these objects.

For all the scenarios listed above (not all were listed, mind you..) the bpy module provides functionality to access and modify data.

Examples

bpy.context

Note
: For the commands below to show the proper output, make sure you have selected object(s) in the 3D view.



```
>>>
active_base  active_bone  active_object  active_pose_bone  area  driver_add(  edit_object  editable_
bones  get(  id_data  is_property_hidden(  is_property_set(  items(  keyframe_insert(  keys(  main
manager  mode  object  particle_edit_object  path_resolve(  path_to_id(  recast_type(  region  rna_t
ype  scene  screen  sculpt_object  selected_bases  selected_bones  selected_editable_bases  selected
_editable_bones  selected_editable_objects  selected_objects  selected_pose_bones  space_data  textu
re_paint_object  tool_settings  user_preferences  values(  vertex_paint_object  visible_bones  visib
le_pose_bones  weight_paint_object  window
>>> bpy.context.mode
'OBJECT'

>>> bpy.context.object
[BPy_StructRNA "Object" -> "Cube"]

active_base  active_bone  active_object  active_pose_bone
>>> bpy.context.active_object
[BPy_StructRNA "Object" -> "Cube"]

selected_bases  selected_bones  selected_editable_bases  selected_editable_bones  selected_editable_
objects  selected_objects  selected_pose_bones
>>> bpy.context.selected_objects
[[BPy_StructRNA "Object" -> "Cube"], [BPy_StructRNA "Object" -> "Lamp"], [BPy_StructRNA "Object" ->
"Camera"]]
```

Try it out!

bpy.context.mode
: Will print the current 3D View mode (Object, Edit, Sculpt etc.,)

bpy.context.object or **bpy.context.active_object**
: Will give access to the active object in the 3D View

```
 >>> bpy.context.object.location.x = 1
```

Change x location to a value of 1

```
 >>> bpy.context.object.location.x += 0.5
```

Move object from previous x location by 0.5 unit

```
 >>> bpy.context.object.location = [1, 2, 3]
```

Changes x, y, z location

```
 >>> bpy.context.object.location.xyz = [1, 2, 3]
```

```
Same as above
```

```
 >>> type(bpy.context.object.location)
```

```
Data type of objects location
```

```
 >>> dir(bpy.context.object.location)
```

```
Now that is a lot of data that you have access to
```

bpy.context.selected_objects
: Will give access to a list of all selected objects.

```
 >>> bpy.context.selected_objects then press {{Shortcut|Ctrl|Space}}
```

```
 >>> bpy.context.selected_objects[0]
```

```
Prints out name of first object in the list
```

```
 >>> [object for object in bpy.context.selected_objects if object != bpy.context.object]
```

```
Complex one.. But this prints a list of objects not including the active object
```

bpy.data

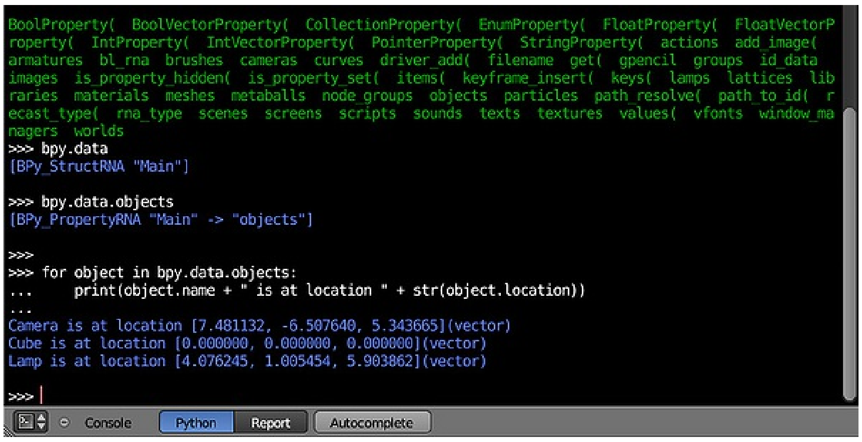
bpy.data has a bunch of functions and variables that give you access to all the data in the Blender file.

You can access following data in the current Blender file:

```
objects, meshes, materials, textures, scenes, screens, sounds, scripts, texts,
cameras, curves, lamps, brushes, armatures, images, lattices, libraries, worlds,
groups, metaballs, particles, node_groups
```

That's a lot of data.

Try it out!



```
BoolProperty(  BoolVectorProperty(  CollectionProperty(  EnumProperty(  FloatProperty(  FloatVectorP
roperty(  IntProperty(  IntVectorProperty(  PointerProperty(  StringProperty(  actions  add_image(
armatures  bl_rna  brushes  cameras  curves  driver_add(  filename  get(  gpencil  groups  id_data
images  is_property_hidden(  is_property_set(  items(  keyframe_insert(  keys(  lamps  lattices  lib
raries  materials  meshes  metaballs  node_groups  objects  particles  path_resolve(  path_to_id(  r
ecast_type(  rna_type  scenes  screens  scripts  sounds  texts  textures  values(  vfonts  window_ma
nagers  worlds
>>> bpy.data
[BPy_StructRNA "Main"]

>>> bpy.data.objects
[BPy_PropertyRNA "Main" -> "objects"]

>>>
>>> for object in bpy.data.objects:
...     print(object.name + " is at location " + str(object.location))
...
Camera is at location [7.481132, -6.507640, 5.343665](vector)
Cube is at location [0.000000, 0.000000, 0.000000](vector)
Lamp is at location [4.076245, 1.005454, 5.903862](vector)

>>> 
```

Exercise

```
 >>> for object in bpy.data.scenes['Scene'].objects: print(object.name)
```

↵ Enter twice

```
Prints the names of all objects belonging to the Blender scene with name "Scene"
```

```
 >>> bpy.data.scenes['Scene'].objects.unlink(bpy.context.active_object)
```

```
Unlink the active object from the Blender scene named 'Scene'
```

```
 >>> bpy.data.materials['Material'].shadows
```

```
 >>> bpy.data.materials['Material'].shadows = False
```

bpy.ops

The tool/action system in Blender 2.5 is built around the concept of operators. These operators can be called directly from console or can be executed by click of a button or packaged in a python script. Very powerful they are..

For a list of various operator categories, click here

Lets create a set of five Cubes in the 3D Viewport. First, delete the existing Cube object by selecting it and pressing X

Try it out!

The following commands are used to specify that the objects are created in layer 1. So first we define an array variable for later reference:

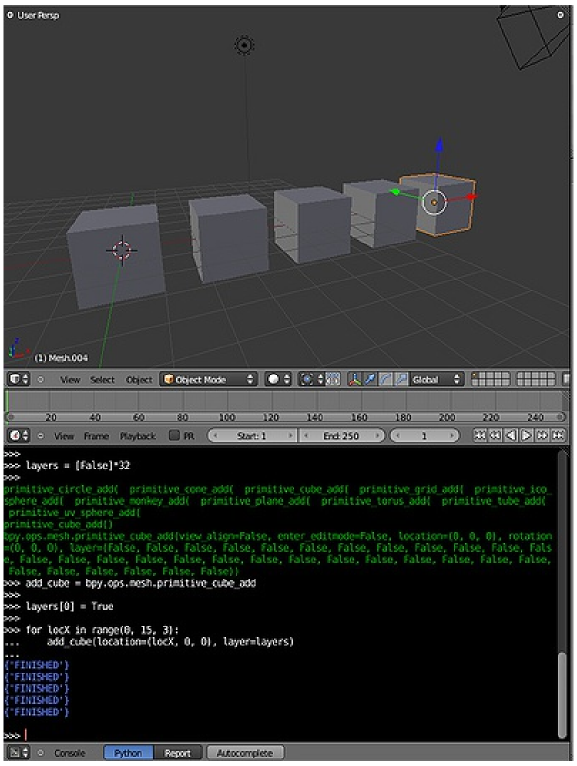
```
>>> mylayers = [False]*20
>>> mylayers[0] = True
```

We create a reference to the operator that is used for creating a cube mesh primitive

```
>>> add_cube = bpy.ops.mesh.primitive_cube_add
```

Now in a for loop, we create the five objects like this (In the screenshot above, I used another method) Press ENTER-KEY twice after entering the command at the shell prompt.

```
>>> for index in range(0, 5):
...     add_cube(location=(index*3, 0, 0), layers=mylayers)
```

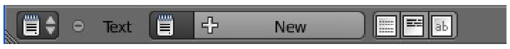


```
>>>
>>> layers = [False]*32
>>>
primitive_circle_add(  primitive_cone_add(  primitive_cube_add(  primitive_grid_add(  primitive_ico_
sphere_add(  primitive_monkey_add(  primitive_plane_add(  primitive_torus_add(  primitive_tube_add(
 primitive_uv_sphere_add(
primitive_cube_add()
bpy.ops.mesh.primitive_cube_add(view_align=False, enter_editmode=False, location=(0, 0, 0), rotation
=(0, 0, 0), layer=(False, False, False, False, False, False, False, False, False, False, False, Fals
e, False, False, False, False, False, False, False, False, False, False, False, False, False, False,
 False, False, False, False, False, False))
>>> add_cube = bpy.ops.mesh.primitive_cube_add
>>>
>>> layers[0] = True
>>>
>>> for locX in range(0, 15, 3):
...     add_cube(location=(locX, 0, 0), layer=layers)
...
{'FINISHED'}
{'FINISHED'}
{'FINISHED'}
{'FINISHED'}
{'FINISHED'}

>>> 
```

The Text Editor

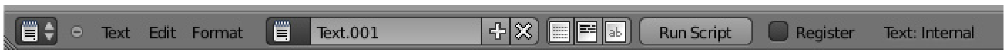
Blender has a Text Editor among its windows types, accessible via the Text Editor button (Text Editor) of the Window type menu, or via ⇧ ShiftF11.

The newly opened Text window is grey and empty, with a very simple toolbar (*Text Toolbar*).



Text Toolbar.

From left to right there are the standard Window type selection button and the window menus. Then there is the Text ID Block browse button followed by the New button for creating new Text files. Once you click it, you will find that the Toolbar has changed.. for good!



Text Toolbar with a file open

Now you find a textbox to change name of your text file, followed by **+** button to create new files. To remove the text block, click the **X** button.

The following three buttons toggle display of line numbers, word-wrap text and syntax highlighting respectively.

Typing on the keyboard produces text in the text buffer. As usual, pressing dragging and releasing LMB selects text.

The following keyboard commands apply:

- CtrlC - Copies the marked text into the text clipboard.
- CtrlX - Cuts out the marked text into the text clipboard.
- CtrlV - Pastes the text from the clipboard at the cursor location in the Text window.
- ⇧ ShiftCtrlAltS - Saves unsaved text as a text file, a File Browser window appears.
- AltS - Saves an already open file.
- AltO - Loads a text, a File Browser window appears.
- AltP - Executes the text as a Python script.
- CtrlZ - Undo.
- Ctrl⇧ ShiftZ - Redo.
- AltR - Reopen (reloads) the current buffer (all non-saved modifications are lost).
- AltM - Converts the content of the text window into 3D text (max 100 chars).

To delete a text buffer just press the X button next to the buffer’s name, just as you do for materials, etc.

The most notable keystroke is AltP which makes the content of the buffer being parsed by the internal Python interpreter built into Blender. The next page will present an example of Python scripting. Before going on it is worth noticing that Blender comes with a fully functional Python interpreter built in, and with a lots of Blender-specific modules, as described in the API references.

The Text Editor has now also some dedicated Python scripts, which add some useful writing tools, like a class/function/variable browser, completion… You can access them through the Text → Text Plugins menu entry.

Other usages for the Text window

The text window is handy also when you want to share your .blend files with the community or with your friends. A Text window can be used to write in a README text explaining the contents of your blender file. Much more handy than having it on a separate application. Be sure to keep it visible when saving! If you are sharing the file with the community and you want to share it under some license you can write the license in a text window.

Demonstration

[video link]

Exercise

Copy the text below in the Text Editor.

```
import bpy
from math import radians, cos, sin

# An object can exist in 20 layers,
# so the following code determines on which layers you want it to be

# Get the cursor's location
cursor = bpy.context.scene.cursor_location

# Radius of the circle
radius = 5

# Space the cubes around the circle. Default is 36 degrees apart
# Get a list of angles converted to radians

anglesInRadians = [radians(degree) for degree in range(0, 360, 36)]

# Loop through the angles, determine x,y using polar coordinates
# and create object
```

```
for theta in anglesInRadians:
    x = cursor.x + radius * cos(theta)
    y = cursor.y + radius * sin(theta)
    z = cursor.z
    bpy.ops.mesh.primitive_cube_add(location=(x, y, z))
```

Execute the script with AltP.

You can see the result of running the above script in this video.

[video link]

Introduction

Since we are working with new and improving Python API, if you have something that needs to be answered, please add it here. We will find answers from dev's if we do not know them and provide an answer here.

Geometry

How can I generate a mesh object using the API?

Download this code example Script_GeneratePyramidMesh.py and run it from the Text Window.

How do I apply a modifier using the API?

```
bpy.ops.object.convert(target='MESH', keep_original=False)
```

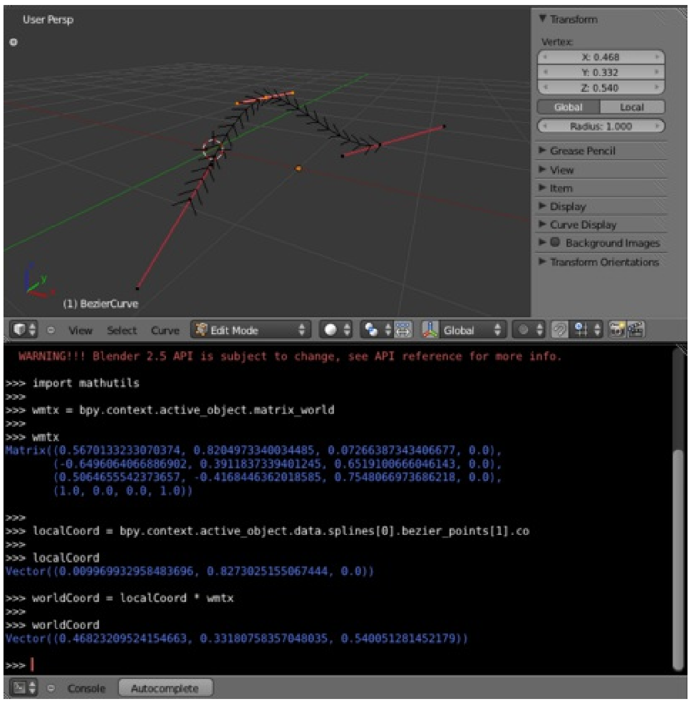
All the modifiers in the stack will be applied.

In case you just want to apply only the subsurf modifier and leave others alone, and create a new mesh (Old mesh will retain all its modifiers), the following code shows one way of doing it.

```
for modifier in bpy.context.object.modifiers:
    if modifier.type != 'SUBSURF':
        modifier.show_render=True
bpy.ops.object.convert(target='MESH', '''keep_original=True''')
```

How do I get the world coordinates of a control vertex of a BezierCurve?

```
wmtx = bpy.context.active_object.matrix_world

localCoord = bpy.context.active_object.data.splines[0].bezier_points[1].co

worldCoord = wmtx * localCoord
```

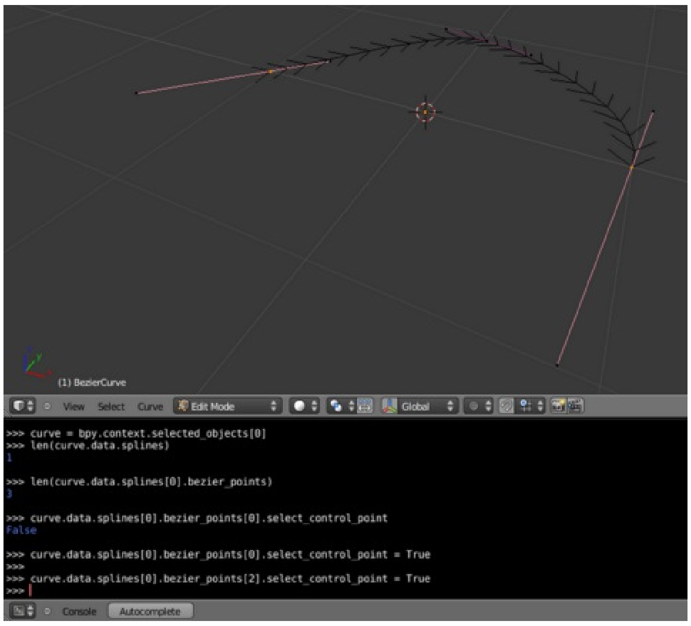


```
 WARNING!!! Blender 2.5 API is subject to change, see API reference for more info.

>>> import mathutils
>>>
>>> wmtx = bpy.context.active_object.matrix_world
>>>
>>> wmtx
Matrix((0.5670133233070374, 0.8204973340034485, 0.07266387343406677, 0.0),
       (-0.6496064066886902, 0.3911837339401245, 0.6519100666046143, 0.0),
       (0.5064655542373657, -0.4168446362018585, 0.7548066973686218, 0.0),
       (1.0, 0.0, 0.0, 1.0))

>>>
>>> localCoord = bpy.context.active_object.data.splines[0].bezier_points[1].co
>>>
>>> localCoord
Vector((0.009969932958483696, 0.8273025155067444, 0.0))

>>> worldCoord = localCoord * wmtx
>>>
>>> worldCoord
Vector((0.46823209524154663, 0.33180758357048035, 0.540051281452179))

>>> 
```

More info...

How do I select/deselect the control points of a Curve

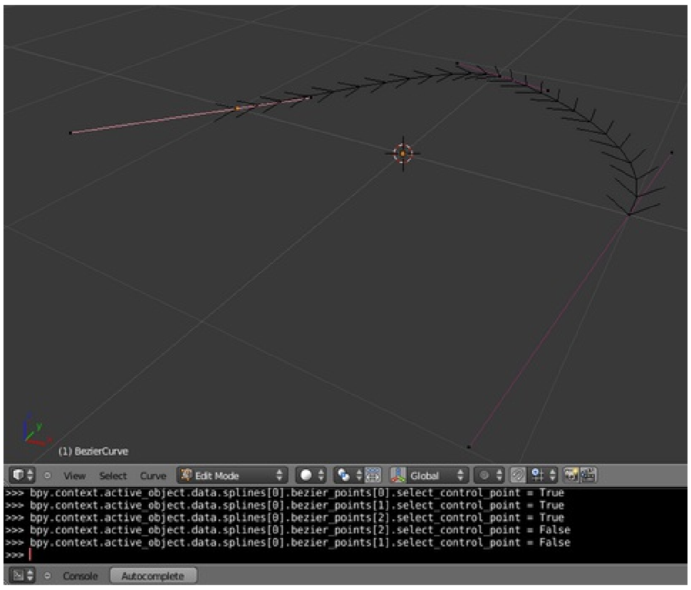
Method 1



```
>>> curve = bpy.context.selected_objects[0]
>>> len(curve.data.splines)
1

>>> len(curve.data.splines[0].bezier_points)
3

>>> curve.data.splines[0].bezier_points[0].select_control_point
False

>>> curve.data.splines[0].bezier_points[0].select_control_point = True
>>>
>>> curve.data.splines[0].bezier_points[2].select_control_point = True
>>>
```

```
curve = bpy.context.selected_objects[0]

curve.data.splines[0].bezier_points[0].select_control_point = True
curve.data.splines[0].bezier_points[2].select_control_point = True
```

Method 2



```
>>> bpy.context.active_object.data.splines[0].bezier_points[0].select_control_point = True
>>> bpy.context.active_object.data.splines[0].bezier_points[1].select_control_point = True
>>> bpy.context.active_object.data.splines[0].bezier_points[2].select_control_point = True
>>> bpy.context.active_object.data.splines[0].bezier_points[2].select_control_point = False
>>> bpy.context.active_object.data.splines[0].bezier_points[1].select_control_point = False
>>>
```

```
bpy.context.active_object.data.splines[0].bezier_points[0].select_control_point = True
```

More info...

Materials

How to link a mesh/object to a material?

TODO

Customization

How do I automate custom hotkeys?

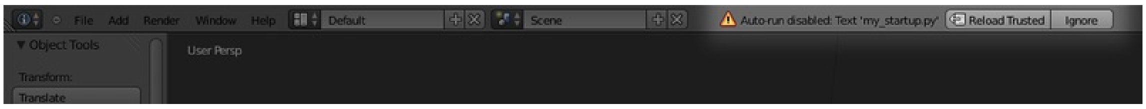
Scripting & Security

The ability to include Python scripts within blend files is valuable for advanced tasks such as rigging, automation and using the game-engine, however it poses a security risk since Python doesn't restrict what a script can do.

Therefore, you should only run scripts from sources you know and trust.

Automatic execution is disabled by default, however some blend files need this to function properly.

When a blend file tries to execute a script and is not allowed, a message will appear in the header with the option to **Reload Trusted** or **Ignore** the message.



Auto-run disabled: Text 'my_startup.py'

Scripts in Blend Files

Auto Execution

Here are the different ways blend files may automatically run scripts.

- Registered Text-Blocks
  *A text block can have its **Register** option enabled which means it will load on start.*
- Animation Drivers
  *Python expressions can be used to **drive** values and are often used in more advanced rigs and animations.*
- Game Engine Auto-Start
  scripts are often used for game logic, blend files can have auto-start enabled with runs the game on load.

Manual Execution

There are other ways scripts in a `blend` file may execute that require user interaction (therefor will run even when auto-execution is off), but you should be aware that this is the case since it's not necessarily obvious.

- Running a script in the text editor *(ok, this is obvious!)*.
- Rendering with FreeStyle - *FreeStyle uses scripts to control line styles*
- Running the Game-Engine.

Controlling Script Execution

Blender provides a number of ways to control whether scripts from a blend file are allowed to automatically execute.

First of all, the file-selector has the option **Trusted Source** which you can use on a case-by-case basis to control auto-execution.

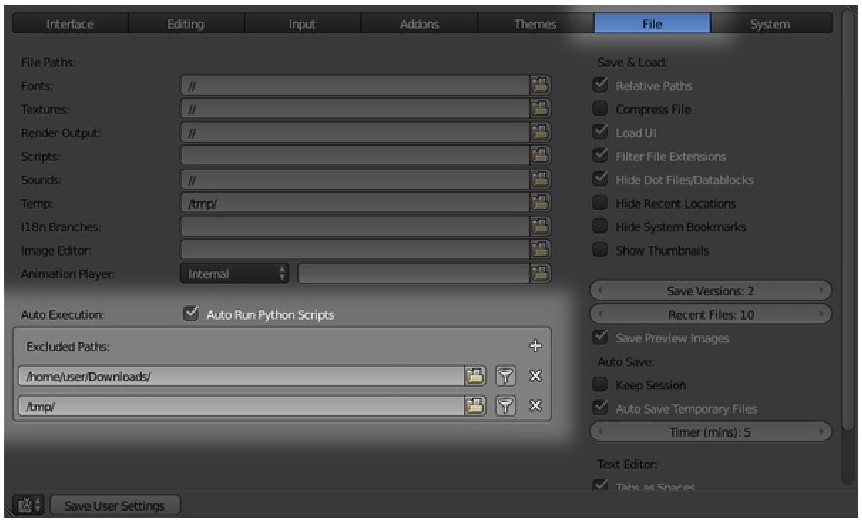
However you may forget to set this, or open a file without going through the file selector - so you can change the default (described next).

Setting Defaults

In the **File** section of the user-preferences there is the toggle **Auto-Run Python Scripts**.

This means the **Trusted Source** option in the file-selector will be enabled by default, and scripts can run when blend files are loaded without using the file selector.

Once enabled you have the option to exclude certain directories, a typical configuration would be to trust all paths except for the download directory.



File Paths:
Fonts: //
Textures: //
Render Output: //
Scripts:
Sounds: //
Temp: /tmp/
I18n Branches:
Image Editor:
Animation Player: Internal

Auto Execution: [✓ Auto Run Python Scripts]
Excluded Paths:
/home/user/Downloads/
/tmp/

Save & Load:
[✓ Relative Paths]
Compress File
[✓ Load UI]
[✓ Filter File Extensions]
[✓ Hide Dot Files/Datablocks]
Hide Recent Locations
Hide System Bookmarks
Show Thumbnails
Save Versions: 2
Recent Files: 10
[✓ Save Preview Images]
Auto Save:
Keep Session
[✓ Auto Save Temporary Files]
Timer (mins): 5
Text Editor:
[✓ Tabs as Spaces]

Command Line

You may want to perform batch rendering or some other task from the command line - running Blender without an interface.

In this case the user-preferences are still used but you may want to override them.

- Enable with `-y` or `--enable-autoexec`
- Disable with `-Y` or `--disable-autoexec`

Example - rendering an animation in background mode, allowing drivers and other scripts to run:

```
blender --background --enable-autoexec my_movie.blend --render-anim
```

Note: these command line arguments can be used to start a regular blender instance and will still override the user-preferences.

Blender’s Python API

The full Python API (Application Programmer Interface) of Blender is documented here:

Latest API - may be newer than current stable release!

Specific versions:

2.64 API

2.63 API

2.62 API

2.61 API

2.60a API

2.59 API

2.58 API

2.57 API

2.56 API

Scripts

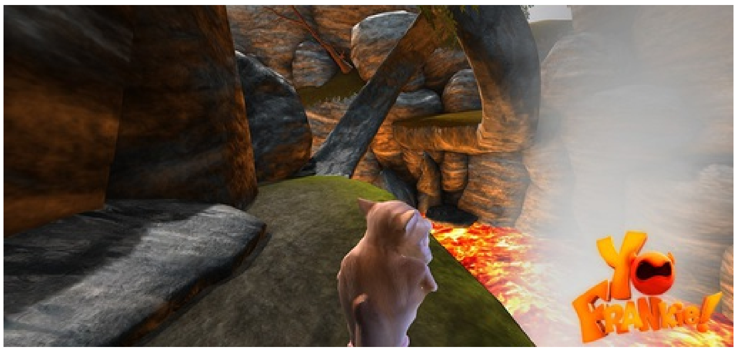
There are more than one hundred different scripts for Blender available on the net.

As with plugins, scripts are very dynamic, changing interface, functionalities and web location fairly quickly, so for an updated list and for a live link to them please refer to one of the two main Blender sites:

- www.blender.org
- www.blenderartists.org
- Python extensions on this wiki.

Introduction to Game Engine

Blender has its own built in Game Engine that allows you to create interactive 3D applications or simulations. The major difference between Game Engine and the conventional Blender system is in the rendering process. In the normal Blender engine, images and animations are built off-line – once rendered they cannot be modified. Conversely, the Blender Game Engine renders scenes continuously in real-time, and incorporates facilities for user interaction during the rendering process.



Screenshot from "Yo Frankie", produced with Blender Game Engine

The Blender Game Engine oversees a game loop, which processes logic, sound, physics and rendering simulations in sequential order. The engine is written in C++.

By default, the user has access to a powerful, high level, Event Driven Logic Editor which is comprised of a seriers of specialised components called "Logic Bricks". The Logic Editor provides deep interaction with the simulation, and its functionality can be extended through Python scripting. It is designed to abstract the complex engine features into a simple user interface, which does not require experience with Programming. An overview of the Logic Editor can be found in the Game Logic Screen Layout

The Game Engine is closely integrated with the existing code base of Blender, which permits quick transitions between the traditional modelling featureset and game-specific functionality provided by the program. In this sense, the Game Engine can be efficiently used in all areas of game design, from prototyping to final release.

The Game Engine can simulate content within Blender, however it also includes the ability to export a binary run-time to Windows, Linux and MacOS. There is also basic support for mobile platforms with the Android Blender Player GSOC 2012 project.

There are a number of powerful libraries included in the 2.5 / 2.6 releases of Blender, including:

- Recast - a state of the art navigation mesh construction toolset for games.
- Detour - a path-finding and spatial reasoning toolkit.
- Bullet - a physics engine featuring 3D collision detection, soft body dynamics, and rigid body dynamics
- Audaspace - a sound library for control of audio. Uses OpenAL or SDL

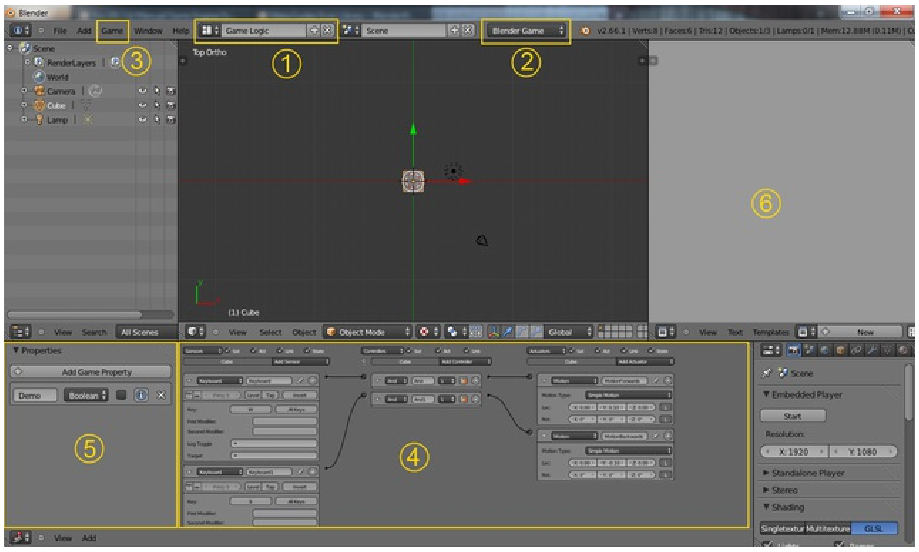
When creating a game or simulation in the BGE, there are four essential steps:

1. Create visual elements that can be rendered. This could be 3D models or images.
2. Enable interaction within the scene using logic bricks to script custom behaviour and determine how it is invoked (using the appropriate "sensors" such as keyboards or joysticks).
3. Create one (or more) camera to give a frustrum from which to render the scene, and modify the parameters to support the environment in which the game will be displayed, such as Stereo rendering.
4. Launch the game, using the internal player or exporting a runtime to the appropriate platform.

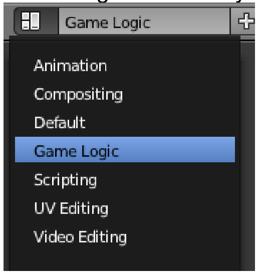
Game Logic Screen Layout

The design, construction, debugging and running of a game utilises a wide range of Blender functions. To help with the process, Blender incorporates a suggested screen layout for setting up BGE games. This includes many already-familiar panels but also a new Logic Editor panel (4) concerned solely with the BGE.

The diagram below shows this default Game Logic screen layout, together with the appropriate options for game setup/debug/running (these should be set up in the order shown).



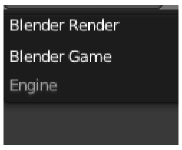
Game Logic Screen Layout



Game Logic Menu

1) Game Logic

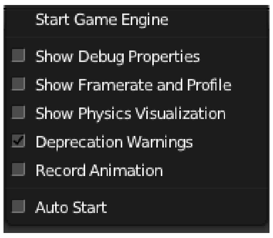
Selected from the list of screen layouts for various applications. This includes many already-familiar panels Information, 3D view, Properties but also a new Logic Editor panel concerned solely with the BGE.



Render Engine Menu

2) Blender Game

Selected from the render engine menu. This specifies that all output will be output by the real-time Blender Game Engine renderer. It also opens various other menu options such as the Game options (see below) and a range of Properties for the BGE renderer properties (see below)



Game Options

3) Game

This menu gives various options for conditions for running the Game Engine.

Note that this menu is only available when the render engine is set to Blender Game.

Start Game: Run game in Game Engine (shortcut p or ⇧ ShiftP when the mouse cursor is over the 3D View window).
Show Debug Properties: Show properties marked for debugging while game runs
Show framerate and profile :Show framerate and profiling information while game runs
Show Physics visualization: show a vizualisation of physics bounds and interactions
Deprecation warnings : Print warnings when using deprecated features in the python API
Record animation : Record animation to F-curves
Auto Start : Automatically start game at load time

4) Logic Editor panel

The Logic Editor is where the logic, properties and states are set up to control the behaviour of the objects in the game. (The Logic Editor panel can also be displayed by selecting Logic Editor in the Display Editor menu, by pressing ⇧ ShiftF2, or by pressing Ctrl→).

5) Properties

Two Meanings for the Same Word

Note that the name "Property" has two different uses in Blender terminology - firstly in the wider use of the Property Display Panel as described here, and secondly as the term used for specific Game Engine logic variables which are also called "properties".

The Property panel of the screen is selected as usual from the main Information menu. However note that several sections of the Property panel are changed when the render engine (2) is changed from Blender Render to Blender Game.

See following sections for details of the content of Physics Properties panels.

Logic, Properties and States

Game Logic is the default scripting layer in the game engine. Each GameObject in the game may store a collection of logical components (Logic Bricks) which control its behavior within the scene. Logic bricks can be combined to perform user-defined actions that determine the progression of the simulation.

Logic Bricks

The main part of game logic can be set up through a graphical interface the Logic Editor, and therefore does not require detailed programming knowledge. Logic is set up as blocks (or "bricks") which represent preprogrammed functions; these can be tweaked and combined to create the game/application. There are three types of logic brick: Sensors, Controllers and Actuators. Sensors are primitive event listeners, which are triggered by specific events, such as a collision, a key press or mouse movement. Controllers carry out logic operations on sensor output, and trigger connected actuators when their operating conditions are met. Actuators interact with the simulation directly, and are the only components in the game which are able to do so (other than the Python controller, and other simulation components such as Physics

Properties

Properties are like variables in other programming languages. They are used to save and access data values either for the whole game (eg. scores), or for particular objects/players (e.g. names). However, in the Blender Game Engine, a property is associated with an object. Properties can be of different types, and are set up in a special area of the Logic Editor.

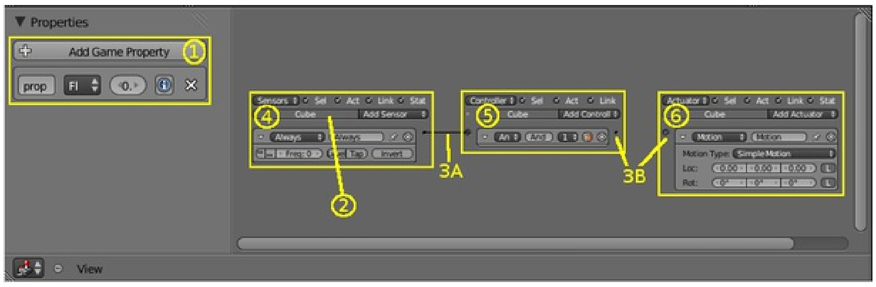
States

Another useful feature is object States. At any time while the simulation is running, the object will process any logic which belongs to the current state of the object. States can be used to define groups of behaviour - eg. an actor object may be "sleeping", "awake" or "dead", and its logic behavior may be different in each of these three states. The states of an object are set up, displayed and edited in the Controller logic bricks for the object.

Logic Editor

The Logic Editor provides the main method of setting up and editing the game logic for the various actors (i.e. objects) that make up the game. The logic for the objects which are currently selected in the associated 3D panel are displayed as logic bricks, which are shown as a table with three columns, showing sensors, controllers, and actuators, respectively. The links joining the logic bricks conduct the pulses between sensor-controller and controller-actuator.

To give you a better understanding of the Logic Editor panel, the image below shows a typical panel content in which the major components have been labeled. We will look at each one individually.



The different parts of the Logic Panel.

1) Game Property Area

Game properties are like variables in other programming languages. They are used to save and access data associated with an object. Several types of properties are available. Properties are declared by clicking the Add Game Property button in this area. For a more in-depth look at the content, layout and available operations in this area, see Properties.

2) Object Name

This box shows the name of the object which owns the logic bricks below.

3) Links

Links (3A) indicate the direction of logical flow between objects. Link lines are drawn by LMB dragging from one Link node (3B) to another. Links can only be drawn from Sensors to Controllers, or from Controllers to Actuators. You cannot directly link Sensors to Actuators; likewise, Actuators cannot be linked back to Sensors (however special actuator and sensor types are available to provide these connections).

Sending nodes (the black circles found on the right-hand side of Sensors and Controllers) can send to multiple Reception nodes (the white circles found on the left-hand side of Controllers and Actuators). Reception nodes can likewise receive multiple links.

Links can be created between logic bricks belonging to different objects.

To delete a link between two nodes, LMB drag between the two nodes.

4) Sensor Area

This column contains a list of all sensors owned by the active object (and any other selected objects). New sensors for the active object are created using the "Add Sensor" button. For a more in-depth look at the content, layout and available operations in this area, see Sensors.

5) Controller Area

This column contains a list of all controllers owned by the active object (and any other selected objects). New controllers for the active object are created using the "Add Controller" button, together with the creation of states for the active object. For a more in-depth look at the content, layout, and available operations in this area, see Controllers.

6) Actuator Area This column contains a list of all actuators owned by the active object (and any other selected objects). New actuators for the active object are created using the "Add Actuator" button. For a more in-depth look at the content, layout, and available operations in this area, see Actuators.

Sensors

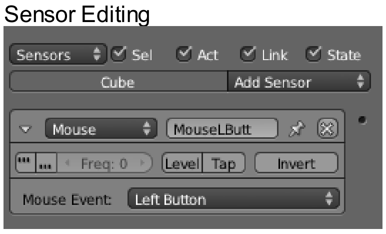
Sensors are the logic bricks that cause the logic to do anything. Sensors give an output when something happens, e.g. a trigger event such as a collision between two objects, a key pressed on the keyboard, or a timer for a timed event going off. When a sensor is triggered, a positive pulse is sent to all controllers that are linked to it.

The logic blocks for all types of sensor may be constructed and changed using the Logic Editor; details of this process are given in the Sensor Editing page.

The following types of sensor are currently available:

Actuator	Detects when a particular actuator receives an activation pulse.
Always	Gives a continuous output signal at regular intervals.
Collision	Detects collisions between objects or materials.
Delay	Delays output by a specified number of logic ticks.
Joystick	Detects movement of specified joystick controls.
Keyboard	Detects keyboard input.
Message	Detects either text messages or property values
Mouse	Detects mous events.
Near	Detects objects that move to within a specific distance of themselves.
Property	Detects changes in the properties of its owner object.
Radar	Detects objects that move to within a specific distance of themselves, within an angle from an axis.
Random	Generates random pulses.
Ray	Shoots a ray in the direction of an axis and detects hits.
Touch	Detects when the object is in contact with another object.

Sensor Editing

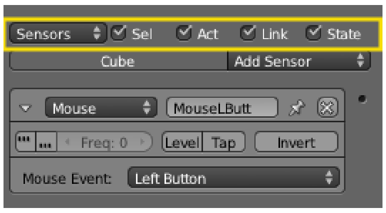


Sensor Column with Typical Sensor

Blender sensors can be set up and edited in the left-hand column of the Logic Panel. This page describes the general column controls, and also those parameters which are common to all individual sensor types.

The image shows a typical sensor column with a single example sensor. At the top of this column, the column heading includes menus and buttons to control which of all the sensors in the current Game Logic are displayed.

Column Heading



Sensor Column Heading

The column headings contain controls to set which sensors, and the level of detail given, in the sensor column. This is very useful for hiding unecessary sensors so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

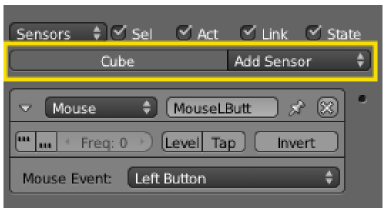
Sensors

Show Objects	Expands all objects.
Hide Objects	Collapses all objects to just a bar with their name.
Show Sensors	Expands all sensors.
Hide Sensors	Collapses all sensors to bars with their names.

It is also possible to filter which sensors are viewed using the four heading buttons:

Sel	Shows all sensors for selected objects.
Act	Shows only sensors belonging to the active object.
Link	Shows sensors which have a link to a controller.
State	Only sensors connected to a controller with active states are shown.

Object Heading



Sensor Object Heading

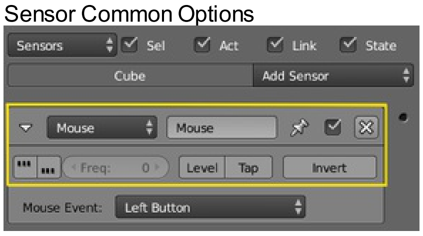
In the column list, sensors are grouped by object. By default, sensors for every selected object appear in the list, but this may be modified by the column heading filters.

At the head of each displayed object sensor list, two entries appear:

Name
: The name of the object.

Add Sensor
: When clicked, a menu appears with the available sensor types. Selecting an entry adds a new sensor to the object. See Sensors for a list of available sensor types.

Sensor Common Options



Common Sensor Options

All sensors have a set of common buttons, fields and menus. They are organized as follows:

Triangle button
: Collapses the sensor information to a single line (toggle).

Sensor type menu
: Specifies the type of the sensor.

Sensor name
: The name of the sensor. This can be selected by the user. It is used to access sensors with Python; it needs to be unique among the selected objects.

Pin button
: Display the sensor even when it is not linked to a visible states controller.

Checkbox button
: Sets active state of the sensor

X **Button**
: Deletes the sensor.

Note about triggers
If a controller does not get trigger by any connected sensor (regardless of the sensors' state) it will not be activated at all.

A sensor triggers the connected controllers on state change. When the sensor changes its state from negative to positive or positive to negative, the sensor triggers the connected controllers. A sensor triggers a connected controller as well when the sensor changes from deactivation to activation.

The following parameters specifies how the sensor triggers connected controllers:

True level triggering. If this is set, the connected controllers will be triggered as long as the sensor's state is positive. The sensor will trigger with the delay (see parameter: frequency) of the sensor.

False level triggering. If this is set, the connected controllers will be triggered as long as the sensor's state is negative. The sensor will trigger with the delay (see parameter: frequency) of the sensor.

Freq
: Despite it's name "Frequency", this parameter sets the delay between repeated triggers, measured in frames (also known as logic ticks). The default value is 0 and it means no delay. It is only used at least one of the level triggering parameters are enabled.
Raising the value of freq is a good way for saving performance costs by avoiding to execute controllers or activate actuators more often than necessary.

Examples: (Assuming the default frame rate with a frequency of 60 Hz (60 frames per second)).

freq	meaning	frames with trigger	frames without trigger	period in frames	frequency in frames/sec
0	The sensor triggers the next frame.	1	0	1	60
1	The sensor triggers at one frame and waits another one until it triggers again. It results in half speed.	1	1	2	30
29	The sensor triggers one frame and waits 29 frames until it triggers again.	1	29	30	2
59	The sensor triggers one frame and waits 59 frames until it triggers again.	1	59	30	1

Level Button
: Triggers connected controllers when state (of the build-in state machine) changes. (For more information see States).

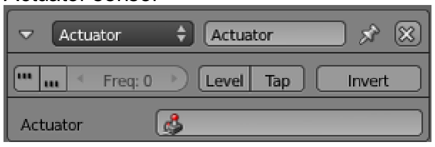
The following parameters specifies how the sensor's status gets evaluated:

Tap Button
: Changes the sensor's state to to negative one frame after changing to positive even if the sensor evaluation remains positive. As this is a state change it triggers the connected controllers as well. Only one of Tap or Level can be activated.
If the *TRUE level triggering* is set, the sensor state will consecutive change from True to False until the sensor evaluates False.

The *FALSE level triggering* will be ignored when the *Tap* parameter is set.

Invert Button
This inverts the sensor output.
If this is set, the sensor's state will be inverted. This means the sensors's state changes to positive when evaluating False and changes to False when evaluating True. If the *Tap* parameter is set, the sensor triggers the controller based on the inverted sensor state.

Actuator sensor



Actuator sensor

The Actuator sensor detects when a particular actuator receives an activation pulse.

The Actuator sensor sends a TRUE pulse when the specified actuator is activated.

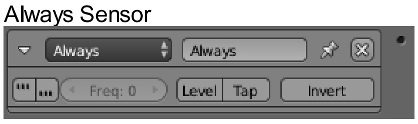
The sensor also sends a FALSE pulse when the specified actuator is deactivated.

See Sensor Common Options for common options.

Special Options:

Actuator
: Name of actuator (NB This must be owned by the same object).

Always Sensor



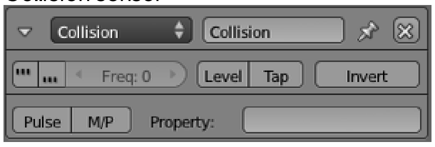
Always sensor

The Always sensor is used for things that need to be done every logic tick, or at every *x* logic tick (with non-null f), or at start-up (with Tap).

See Sensor Common Options for common options.

This sensor doesn’t have any special options.

Collision sensor



Collision sensor

A Collision sensor works like a Touch sensor but can also filter by property or material. Only objects with the property/material with that name will generate a positive pulse upon collision. Leave blank for collision with any object.

See Sensor Common Options for common options.

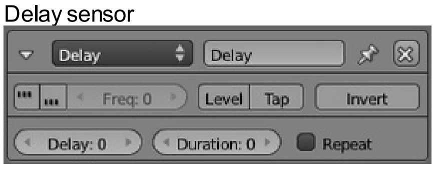
Special Options:

Pulse button
: Makes it sensible to other collisions even if it is still in touch with the object that triggered the last positive pulse.

M/P button
: Toggles between material and property filtering.

Note about soft bodies
The Collision sensor can not detect collisions with soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Delay sensor



Delay sensor

The Delay sensor is designed for delaying reactions a number of logic ticks. This is useful if an other action has to be done first or to time events.

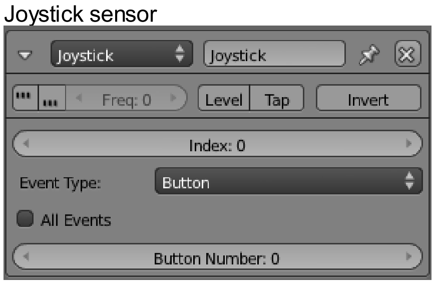
See Sensor Common Options for common options. Special Options:

Delay
: The number of logic ticks the sensor waits before sending a positive pulse.

Duration
: The number of logic ticks the sensor waits before sending the negative pulse.

Repeat Button
: Makes the sensor restart after the delay and duration time is up.

Joystick sensor



Joystick sensor

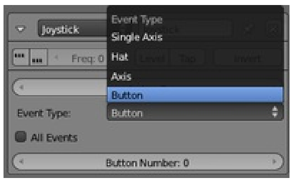
The Joystick sensor triggers whenever the joystick moves. It also detects events on a range of ancilliary controls on the joystick device (hat, buttons, etc.). More than one joystick may be used (see "Index"). The exact layout of the joystick controls will depend on the make and model of joystick used.

See Sensor Common Options for common options.

Special Options:

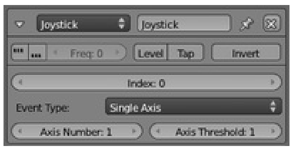
Index
: Specifies which joystick to use.

All Events
: Sensor triggers for all events on this joystick's current type



Joystick Events

Event Type
: A menu to select which joystick event to use

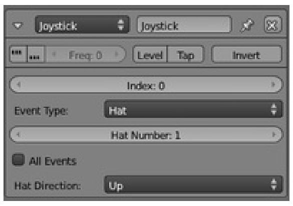


Joystick Single Axis

Single Axis
: Detect movement in a single joystick Axis.

  Axis Number
  : 1 = Horizontal axis (left/right)
  : 2 = Vertical axis (forward/back)
  : 3 = Paddle axis up/down
  : 4 = Joystick axis twist left/right

  Axis Threshold
  : Threshold at which joystick fires (Range 0 - 32768)

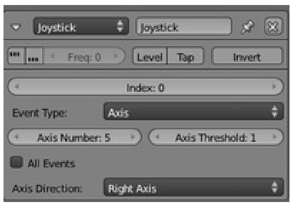


Joystick Hat

Hat
: Detect movement of a specific hat control on the joystick.

  Hat number
  : Specifies which hat to use (max. 2)

  Hat Direction
  : Specifies the direction to use: up, down, left, right, up/right, up/left, down/right, down/left.



Joystick Axis

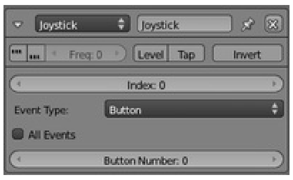
Axis
: Axis Number

  Specifies the axis (1 or 2)

  Axis Threshold
  : Threshold at which joystick fires (Range 0 - 32768)

  Axis Direction specifies the direction to use:

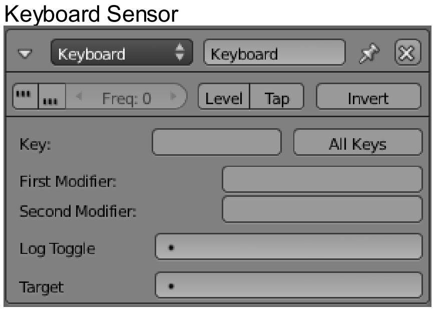
  (Axis Number = 1) Joystick Left, Right, Up, Down
  (Axis Number = 2) Paddle upper (Left); paddle Lower (Right); Joystick twist left (Up) Joystick twist right (Down)



Joystick Button

Button
: Specify the button number to use.

Keyboard Sensor



Keyboard sensor

The Keyboard sensor is for detecting keyboard input. It can also save keyboard input to a String property.

See Sensor Common Options for common options.

Special Options:

Key

This field detects presses on a named key. Press the button with no label and a key to assign that key to the sensor. This is the active key, which will trigger the TRUE pulse. Click the button and then click outside of the button to deassign the key.

A FALSE pulse is given when the key is released.

All keys button

Sends a TRUE pulse when any key is pressed. This is useful for custom key maps with a Python controller.

First Modifier
Second Modifier

Specifies additional key(s), all of which must be held down while the active key is pressed in order for the sensor to give a TRUE pulse. These are selected in the same way as Key. This is useful if you wish to use key combinations, for example CtrlR or ⇧ ShiftAltEsc to do a specific action.

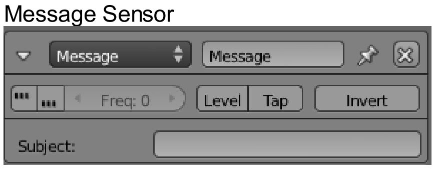
LogToggle

Assigns a Bool property which determines if the keystroke will or will not be logged in the target String. This property needs to be TRUE if you wish to log your keystrokes.

Target

The name of property to which the keystrokes are saved. This property must be of type String. Together with a Property sensor this can be used for example to enter passwords.

Message Sensor



Message sensor

The Message sensor can be used to detect either text messages or property values. The sensor sends a positive pulse once an appropriate message is sent from anywhere in the engine. It can be set up to only send a pulse upon a message with a specific subject.

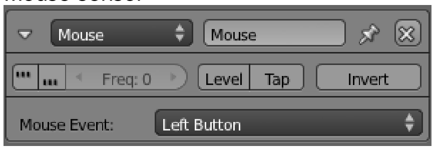
See Sensor Common Options for common options.

Special Options

Subject
: Specifies the message that must be received to trigger the sensor (this can be left blank).

Note: See Message Actuator for how to send messages

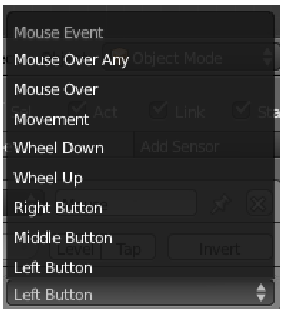
Mouse sensor



Mouse sensor

The Mouse sensor is for detecting mouse events.

See Sensor Common Options for common options.



Mouse Events

Special Options

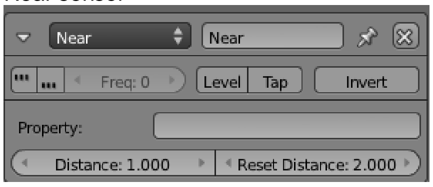
The controller consist only of a list of types of mouse events. These are:

- Mouse over any, gives a TRUE pulse if the mouse moves over any game object.
- Mouse over, gives a TRUE pulse if the mouse moves over the owner object.
- Movement, any movement with the mouse causes a stream of TRUE pulses.
- Wheel Down, causes a stream of TRUE pulses as the scroll wheel of the mouse moves down.
- Wheel Up, causes a stream of TRUE pulses as the scroll wheel of the mouse moves up.
- Right button gives a TRUE pulse.
- Middle button gives a TRUE pulse.
- Left button gives a TRUE pulse.

A FALSE pulse is given when any of the above conditions ends.

There is no logic brick for specific mouse movement and reactions (such as first person camera), these have to be coded in python.

Near sensor



Near sensor

A Near sensor detects objects that move to within a specific distance of themselves. It can filter objects with properties, like the Collision sensor.

See Sensor Common Options for common options.

Special Options

Property

This field can be used to limit the sensor to look for only those objects with this property.

Distance

The number of blender units it will detect objects within.

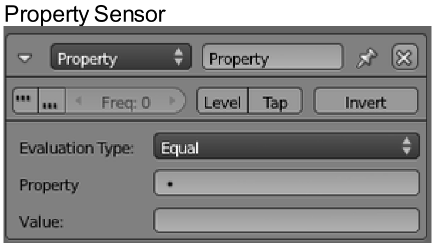
Reset

The distance the object needs to be to reset the sensor (send a FALSE pulse).

Notes
1) The Near sensor can detect objects "through" other objects (walls etc).
2) Objects must have "Actor" enabled to be detected.

Note about soft bodies
The Near sensor can not detect soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Property Sensor



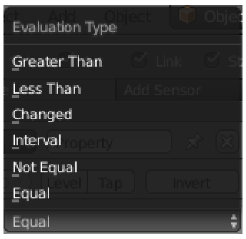
Evaluation Type: Equal
Property: •
Value:

Property sensor

The Property sensor detects changes in the properties of its owner object.

See Sensor Common Options for common options.

Special Options



Property Evaluation

Evaluation Type

Specifies how the property will be evaluated against the value(s).

Greater Than
: Sends a TRUE pulse when the property value is greater than the Value in the sensor.

Less Than
: Sends a TRUE pulse when the property value is less than the Value in the sensor.

Changed
: Sends a TRUE pulse as soon as the property value changes.

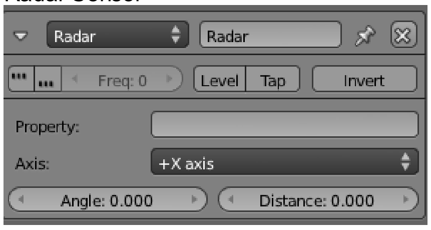
Interval
: Sends a TRUE pulse when the Value of the property is between the Min and Max values of the sensor.

Not Equal
: Sends a TRUE pulse when the property value differs from the Value in the sensor.

Equal
: Sends a TRUE pulse when the property value matches the Value in the sensor.

Note the names of other properties can also be entered to compare properties.

Radar Sensor



Radar sensor

The Radar sensor works much like a Near sensor, but only within an angle from an axis, forming an invisible cone with the top in the objects' center and base at a distance on an axis.

See Sensor Common Options for common options.

Special Options

Property
: This field can be used to limit the sensor to look for only those objects with this property.

Notes
1) The Radar sensor can detect objects "through" other objects (walls etc).
2) Objects must have "Actor" enabled to be detected.

Axis
: This menu determines the direction of the radar cone. The ± signs is whether it is on the axis direction (+), or the opposite (-).

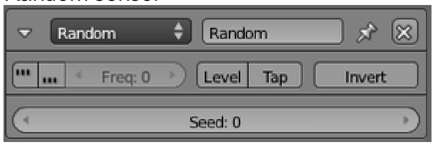
Angle
: Determines the angle of the cone. (Range: 0.00 to 179.9 degrees).

Distance
: Determines the length of the cone. (Blender units).

This sensor is useful for giving bots sight only in front of them, for example.

Note about soft bodies
The Radar sensor can not detect soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Random sensor



Random sensor

The Random sensor generates random pulses.

See Sensor Common Options for common options.

Special Options:

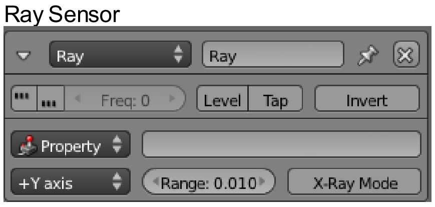
Seed

This field to enter the initial seed for the random number algorithm. (Range 0-1000).

Notes -

1) 0 is not random, but is useful for testing and debugging purposes.
2) If you run several times with the same Seed, the sequence of intervals you get will be the same in each run, although the intervals will be randomly distibuted.

Ray Sensor



Ray sensor

The Ray sensor shoots a ray in the direction of an axis and sends a positive pulse once it hits something. It can be filtered to only detect objects with a given material or property.

See Sensor Common Options for common options.

Special Options: It shares a lot of buttons and fields with Radar sensor.

Property
This field can be used to limit the sensor to look for only those objects with this property.

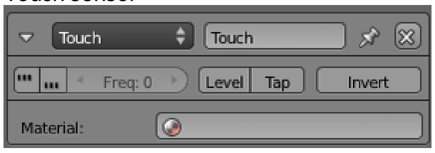
Notes
1) Unless the Property field is set, the Ray sensor can detect objects "through" other objects (walls etc).
2) Objects must have "Actor" enabled to be detected.

Axis
This menu determines the direction of the ray. The ± signs is whether it is on the axis direction (+), or the opposite (-).

Range
Determines the length of the ray. (Blender units).

X-Ray Mode button
Makes it x-ray, so that it sees through objects that don’t have the property or material specified in the filter field.

Touch sensor



Touch sensor

The Touch sensor sends a positive pulse when the object is in contact with another object.

See Sensor Common Options for common options.

Special Options

Material

This field is for filtering materials. Only contact with the material in this field will generate a positive pulse. Leave blank for touch with any object.

A TRUE pulse is sent on collision and the FALSE pulse is sent once the objects are no longer in contact.

Touch sensor has been removed in 2.69

The Touch sensor is no longer available in v2.69 or later. The Collision Sensor now provides the same functionality.

Note about soft bodies

The Touch sensor can not detect collisions with soft bodies. This is a limitation in Bullet, the physics library used by the Game Engine.

Controllers

The controllers are the bricks that collect data sent by the sensors, and also specify the state for which they operate. After performing the specified logic operations, they send out pusle signals to drive the actuators to which they are connected.

When a sensor is activated, it sends out a positive pulse, and when it is deactivated, it sends out a negative pulse. The controllers' job is to check and combine these pulses to trigger the proper response.

The logic blocks for all types of controller may be constructed and changed using the Logic Editor; details of this process are given in the Controller Editing page.

Controller Types

There are eight types of controller logic brick to carry out the logic process on the input signal(s): these are described in the separate pages shown below:

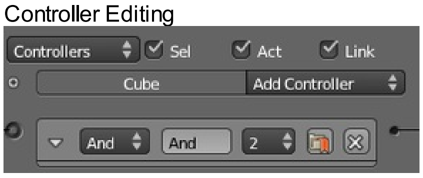
- AND
- OR
- XOR
- NAND
- NOR
- XNOR
- Expression
- Python

This table gives a quick overview of the logic operations performed by the logical controller types. The first column, input, represents the number of positive pulses sent from the connected sensors. The following columns represent each controller's response to those pulses. True means the conditions of the controller are fulfilled, and the actuators it is connected to will be activated; false means the controller's conditions are not met and nothing will happen. Please consult the individual controller pages for a more detailed description of each controller.

Note
It is assumed that more than one sensor is connected to the controller. For only one sensor, consult the "All" line.

Positive sensors	**Controllers**					
	AND	OR	XOR	NAND	NOR	XNOR
None	False	False	False	True	True	True
One	False	True	True	True	False	False
Multiple, not all	False	True	False	True	False	True
All	True	True	False	False	False	True

Controller Editing

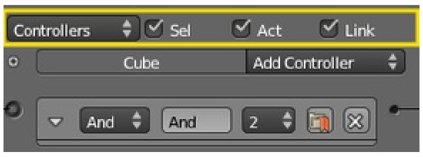


Controller Column with Typical Sensor

Blender controllers can be set up and edited in the central column of the Logic Panel. This page describes the general column controls, those parameters which are common to all individual controller types, and how different states for the objects in the logic system can be set up and edited.

The image shows a typical controller column with a single controller. At the top of this column, and for sensors and actuators, the column heading includes menus and buttons to control which of all the controllers in the current Game Logic are displayed.

Column Heading



Controller Column Headings

The column headings contain controls to set which controllers appear, and the level of detail given, in the controller column. This is very useful for hiding unecessary controllers so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

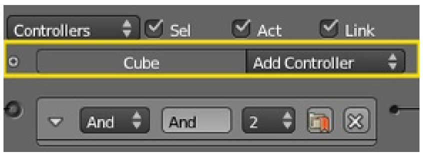
Controllers

Show Objects	Expands all objects.
Hide Objects	Collapses all objects to just a bar with their name.
Show Controllers	Expands all Controllers.
Hide Controllers	Collapses all Controllers to bars with their names.

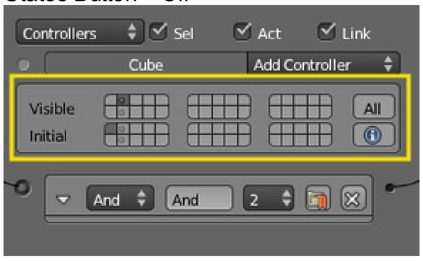
It is also possible to filter which controllers are viewed using the three heading buttons:

Sel	Shows all controllers for selected objects.
Act	Shows only controllers belonging to the active object.
Link	Shows controllers which have a link to actuators/sensors.

Object Heading



Controller Column Object Headings, Used States Button = Off



Controller Column Object Headings, Used States Button = On

In the column list, controllers are grouped by object. By default, controllers for every selected object appear in the list, but this may be modified by the column heading filters.

At the head of each displayed object controller list, three entries appear:

(Used States Button)
Shows which states are in use for the object. Detailed description of the marked panel is given in States.

Name
The name of the object.

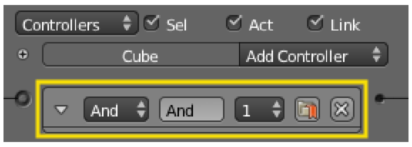
Add Controller
When clicked, a menu appears with the available controller types. Selecting an entry adds a new controller to the object. See Controllers for a list of available controller types.

AND Controller

This controller gives a positive (TRUE) output when

All its inputs are TRUE, and
The object is in the designated State.
For all other conditions the controller gives a negative (FALSE) output.

Options:



AND Controller

Controller Type menu
: Specifies the type of the controller.

Controller Name
: The name of the controller. This can be selected by the user. It is used to access controllers with python; it needs to be unique among the selected objects.

State Index
: Sets the designated state for which this controller will operate.

Preference Button
: If on, this controller will operate before all other non-preference controllers (useful for start-up scripts).

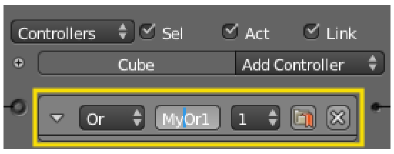
X **Button**
: Deletes the sensor.

OR Controller

This controller gives a positive (TRUE) output when

Any one or more of its inputs are TRUE, and
The object is in the designated State.
For all other conditions the controller gives a negative (FALSE) output.

Options:



OR Controller

Controller Type menu
Specifies the type of the controller.

Controller Name
The name of the controller. This can be selected by the user. It is used to access controllers with python; it needs to be unique among the selected objects.

State Index
Sets the designated state for which this controller will operate.

Preference Button
If on, this controller will operate before all other non-preference controllers (useful for start-up scripts).

X **Button**
Deletes the sensor.

NAND Controller

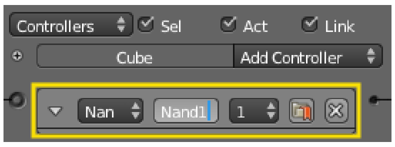
This controller **activates** all connected actuators if

- the game object is in the designated state
- at least one connected sensor triggers the controller
- at least one connected sensor evaluated False

This controller **deactivates** all connected actuators if

- the game object is in the designated state
- at least one connected sensor triggers the controller
- ALL connected sensor evaluated True

Options:



NAND Controller

Controller Type menu
: Specifies the type of the controller.

Controller Name
: The name of the controller. This can be selected by the user. It is used to access controllers with python; it needs to be unique among the selected objects.

State Index
: Sets the designated state for which this controller will operate.

Preference Button
: If enabled, this controller will operate before all other non-preference controllers (useful for start-up scripts).

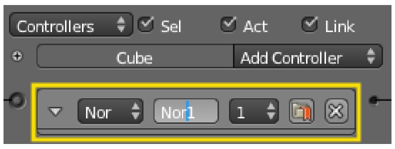
X **Button**
: Deletes the sensor.

NOR Controller

This controller gives a positive (TRUE) output when

None of its inputs are TRUE, and
The object is in the designated State.
For all other conditions the controller gives a negative (FALSE) output.

Options:



NOR Controller

Controller Type menu
Specifies the type of the controller.

Controller Name
The name of the controller. This can be selected by the user. It is used to access controllers with python; it needs to be unique among the selected objects.

State Index
Sets the designated state for which this controller will operate.

Preference Button
If on, this controller will operate before all other non-preference controllers (useful for start-up scripts).

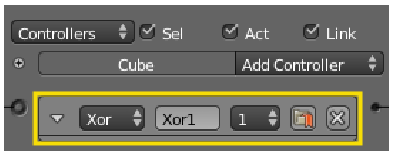
X **Button**
Deletes the sensor.

XOR Controller

This controller gives a positive (TRUE) output when

One (and only one) of its inputs are TRUE, and
The object is in the designated State.
For all other conditions the controller gives a negative (FALSE) output.

Options:



XOR Controller

Controller Type menu
Specifies the type of the controller.

Controller Name
The name of the controller. This can be selected by the user. It is used to access controllers with python; it needs to be unique among the selected objects.

State Index
Sets the designated state for which this controller will operate.

Preference Button
If on, this controller will operate before all other non-preference controllers (useful for start-up scripts).

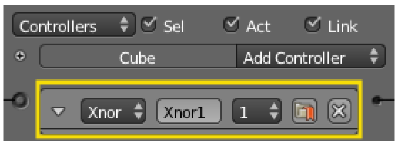
X **Button**
Deletes the sensor.

XNOR Controller

This controller gives a positive (TRUE) output when

One (and only one) of its inputs are FALSE, and
The object is in the designated State.
For all other conditions the controller gives a negative (FALSE) output.

Options:



XNOR Controller

Controller Type menu
Specifies the type of the controller.

Controller Name
The name of the controller. This can be selected by the user. It is used to access controllers with python; it needs to be unique among the selected objects.

State Index
Sets the designated state for which this controller will operate.

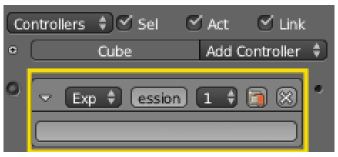
Preference Button
If on, this controller will operate before all other non-preference controllers (useful for start-up scripts).

X **Button**
Deletes the sensor.

Expression Controller

This controller evaluates a user written expression, and gives a positive (TRUE) output when

The result of the expression is TRUE, and
The object is in the designated State.
For all other conditions the controller gives a negative (FALSE) output.



Expression Controller

Expression
The expression, which is written in the box, can consist of variables, constants and operators. These must follow the rules laid out below.

Variables

You can use:

- **sensors names**,
- **properties**: assign a game property to an object and use it in a controller expression.

These cannot contain blank spaces.

Operations

Mathematical operations

Operators: *, /, +, -

Returns: a number

Examples: 3 + 2, 35 / 5

Logical operations

- Comparison operators: <, >, >=, <=, ==, !=
- Booleans operators: AND, OR, NOT

Returns: True or False.

Examples: 3 > 2 (True), 1 AND 0 (False)

Conditional statement (if)

Use:

```
if( expression, pulse_if_expression_is_true, pulse_if_expression_is_false )
```

If the controller evaluates **`expression`** to True:

- if **`pulse_if_expression_is_true`** is `True`, the controller sends a positive pulse to the connected actuators.
- if **`pulse_if_expression_is_true`** is `False`, the controller sends a negative pulse to the connected actuators.

If the controller evaluates **`expression`** to False:

- if **`pulse_if_expression_is_false`** is `True`, the controller sends a positive pulse to the connected actuators.
- if **`pulse_if_expression_is_false`** is `False`, the controller sends a negative pulse to the connected actuators.

Examples

Given the object has a property **`coins`** equal to 30:

```
coins > 20
```

returns True (the controller sends a positive pulse to the connected actuators).

Given the object has:

- a sensor called **`Key_Inserted`** equal to True,
- a property named **`Fuel`** equal to False,

```
Key_Inserted AND Fuel
```

returns False (the controller sends a negative pulse to the connected actuators).

This is the same as doing:

```
if (Key_Inserted AND Fuel, True, False)
```

Instead, you could do:

```
if (Key_Inserted AND Fuel, False, True)
```

to return a positive pulse when **`Key_Inserted AND Fuel`** returns False.

You can also do:

```
if ((Key_Inserted AND Fuel) OR (coins > 20), True, False)
```

This expression returns True, hence in this case the controller sends a positive pulse to the connected actuators.

Python Controller

Actuators

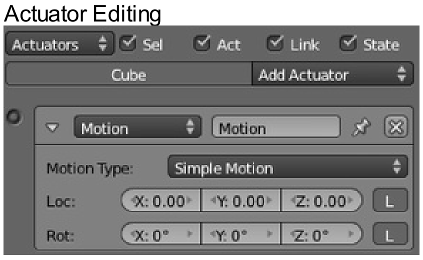
Actuators perform actions, such as move, create objects, play a sound. The actuators initiate their functions when they get a positive pulse from one (or more) of their controllers.

The logic blocks for all types of actuator may be constructed and changed using the Logic Editor; details of this process are given in the Actuator Editing page.

The following types of actuator are currently available:

Action	Handles armature actions. This is only visible if an armature is selected.
Camera	Has options to follow objects smoothly, primarily for camera objects, but any object can use this.
Constraint	Constraints are used to limit object's locations, distance, or rotation. These are useful for controlling the physics of the object in game.
Edit Object	Edits the object's mesh, adds objects, or destroys them. It can also change the mesh of an object (and soon also recreate the collision mesh).
Filter 2D	Filters for special effects like sepia colours or blur.
Game	Handles the entire game and can do things as restart, quit, load, and save.
Message	Sends messages, which can be received by other objects to activate them.
Motion	Sets object into motion and/or rotation. There are different options, from "teleporting" to physically push rotate objects.
Parent	Can set a parent to the object, or unparent it.
Property	Manipulates the object's properties, like assigning, adding, or copying.
Random	Creates random values which can be stored in properties.
Scene	Manage the scenes in your .blend file. These can be used as levels or for UI and background.
Sound	Used to play sounds in the game.
State	Changes states of the object.
Steering	Provides pathfinding options for the object.
Visibility	Changes visibility of the object.

Actuator Editing

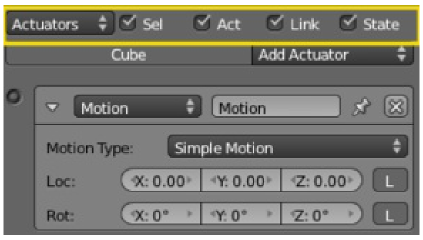


Actuator Column with Typical Actuator

Blender actuators can be set up and edited in the right-hand column of the Logic Panel. This page describes the general column controls, and also those parameters which are common to all individual actuator types.

The image shows a typical actuator column with a single example actuator. At the top of this column, the column heading includes menus and buttons to control which of all the actuators in the current Game Logic are displayed.

Column Heading



Actuator Column Heading

The column headings contain controls to set which actuators, and the level of detail given, in the actuator column. This is very useful for hiding unecessary actuators so that the necessary ones are visible and easier to reach. Both these can be controlled individually.

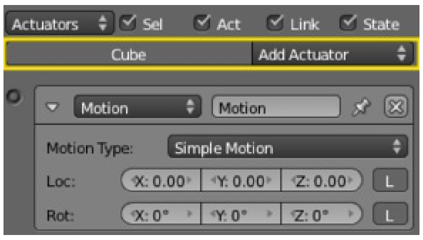
Actuators

Show Objects	Expands all objects.
Hide Objects	Collapses all objects to just a bar with their name.
Show Actuators	Expands all actuators.
Hide Actuators	Collapses all actuators to bars with their names.

It is also possible to filter which actuators are viewed using the four heading buttons:

Sel	Shows all actuators for selected objects.
Act	Shows only actuators belonging to the active object.
Link	Shows actuators which have a link to a controller.
State	Only actuators connected to a controller with active states are shown.

Object Heading



Actuator Object Heading

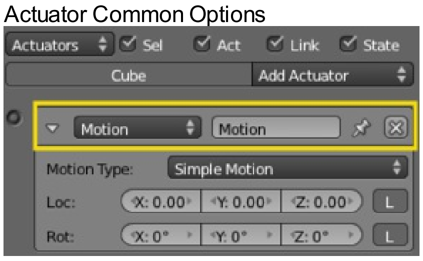
In the column list, actuators are grouped by object. By default, actuators for every selected object appear in the list, but this may be modified by the column heading filters.

At the head of each displayed object sensor list, two entries appear:

Name
The name of the object.

Add
When clicked, a menu appears with the available actuator types. Selecting an entry adds a new actuator to the object. See Actuators for list of available actuator types.

Actuator Common Options



Common Actuator Options

All actuators have a set of common buttons, fields and menus. They are organized as follows:

Triangle button
: Collapses the sensor information to a single line (toggle).

Actuator type menu
: Specifies the type of the sensor.

Actuator name
: The name of the actuator. This can be selected by the user. It is used to access actuators with python; it needs to be unique among the selected objects.

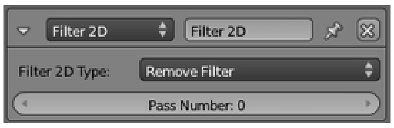
X **Button**
: Deletes the actuator.

Filter 2D Actuator

2D Filters are image filtering actuators, that apply on final render of objects.

Filter 2D Type

Select the type of 2D Filter required.


Filter 2D | Filter 2D
Filter 2D Type: Remove Filter
Pass Number: 0

Edit Object actuator

- Custom Filter
- Invert
- Sepia
- Gray Scale
- Prewitt
- Sobel
- Laplacian
- Erosion
- Dilation
- Sharpen
- Blur
- Motion Blur
- Remove Filter
- Disable Filter
- Enable Filter

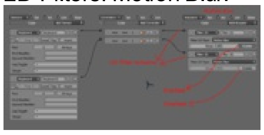
Only one parameter is required for all filters

Pass Number
The pass number for which this filter is to be used.

Details of the filters are given in the descriptive text below.

Motion Blur

Motion Blur is a 2D Filter that needs previous rendering information to produce motion effect on objects. Below you can see Motion Blur filter in Blender window, along with its logic bricks:

2D Filters: Motion Blur.

2D Filters: Game Logic.

To enable this filter:

1. Add appropriate Sensor(s) and Controller(s).
2. Add a 2D Filter Actuator.
3. Select Motion Blur in the drop-down list.
4. Set Motion Blur Value (Factor).

And for disabling this filter:

1. Add appropriate Sensor(s) and Controller(s).
2. Add a 2D Filter Actuator.
3. Select Motion Blur.
4. Toggle Enable button to go to disabled mode.

You can enable Motion Blur filter using a Python controller:

```
from bge import render
render.enableMotionBlur(0.85)
```

And disable it:

```
from bge import render
render.disableMotionBlur()
```

Note
Your graphic hardware and OpenGL driver must support accumulation buffer (`glAccum` function).

Built-In 2D Filters

All 2D filters you can see in 2D Filter actuator have the same architecture, all built-in filters use fragment shader to produce final render view, so your hardware must support shaders.

2D Filters: Motion Blur.

2D Filters: Sepia.

2D Filters: Sobel.

Blur, Sharpen, Dilation, Erosion, Laplacian, Sobel, Prewitt, Gray Scale, Sepia and Invert are built-in filters. These filters can be set to be available in some passes.

To use a filter you should:

1. Create appropriate sensor(s) and controller(s).
2. Create a 2D Filter actuator.
3. Select your filter, for example Blur.
4. Set the pass number that the filter will be applied.

To remove a filter on a specific pass:

1. Create appropriate sensor(s) and controller(s).
2. Create a 2D Filter actuator.
3. Select Remove Filter.
4. Set the pass number you want to remove the filter from it.

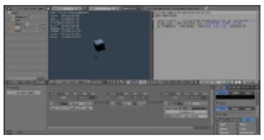
To disable a filter on a specific pass:

1. Create appropriate sensor(s) and controller(s).
2. Create a 2D Filter actuator.
3. Select Disable Filter.
4. Set the pass number you want to disable the filter on it.

To enable a filter on a specific pass:

1. Create appropriate sensor(s) and controller(s)
2. Create a 2D Filter actuator.
3. Select Enable Filter.
4. Set the pass number you want to enable the filter on it.

Custom Filters


2D Filters: Custom Filter.

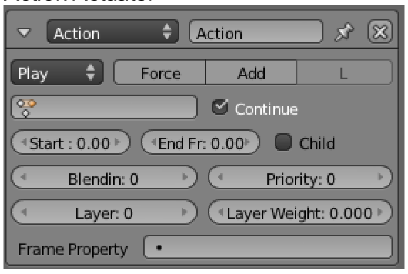
Custom filters give you the ability to define your own 2D filter using GLSL. Its usage is the same as built-in filters, but you must select

Custom Filter in 2D Filter actuator, then write shader program into the Text Editor, and then place shader script name on actuator.

Blue Sepia Example:

```
uniform sampler2D bgl_RenderedTexture;
void main(void)
{
  vec4 texcolor = texture2D(bgl_RenderedTexture, gl_TexCoord[0].st);
  float gray = dot(texcolor.rgb, vec3(0.299, 0.587, 0.114));
  gl_FragColor = vec4(gray * vec3(0.8, 1.0, 1.2), texcolor.a);
}
```

Action Actuator



Action Actuator

Actuates armature actions, and sets the playback method. The Action actuator is only visible when an armature is selected, because actions are stored in the armature.

See Actuator Common Options for common options.

Special Options:

Action Playback Type

Play
: Play ipo once from start to end when a TRUE pulse is received.

Ping Pong
: Play ipo once from start to end when a TRUE pulse is received. When the end is reached play ipo once from end to start when a TRUE pulse is received.

Flipper
: Play ipo once from start to end when a TRUE pulse is received. (Plays backwards when a FALSE pulse is received).

Loop End
: Play ipo continuously from end to start when a TRUE pulse is received.

Loop Start
: Play ipo continuously from start to end when a TRUE pulse is received.

Property
: Uses a property to define what frame is displayed.

Action
: Select the action to use

Continue
: Restore last frame when switching on/off, otherwise play from the start each time.

Start Frame
: Set the start frame of the action.

End Frame
: Set the end frame of the action.

Child Button
: Update action on all children objects as well.

Blendin
: Number of frames of motion blending.

Priority
: Execution priority - lower numbers will override actions with higher numbers. With 2 or more actions at once, the overriding channels must be lower in the stack.

Frame Property
: Assign the action’s current frame number to this property.

Property
: Use this property to define the Action position. Only for Property playback type.

Layer
: The animation layer to play the action on.

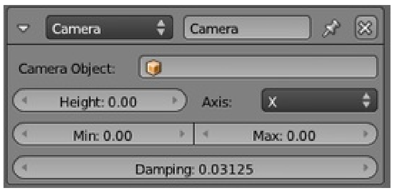
Layer Weight
: How much of the previous layer to blend into this one.

Camera Actuator

Makes the camera follow or track an object.

See Actuator Common Options for common options.

Special Options:



Camera Actuator

Camera Object

Name of the Game Object that the camera follows/tracks.

Height

Height the camera tries to stay above the Game Object's object center

Axis

Axis in which the Camera follows (X or Y)

Min

Minimum distance for the camera to follow the Game Object

Max

Maximum distance for the camera to follow the Game Object

Damping

Strength of the constraint that drives the camera behind the target. Range: 0 to 10. The higher the parameter, the quicker the camera will adjust to be inside the constrained range (of min, max and height).

Constraints Actuator

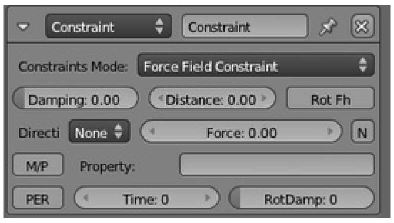
Adds a constraint to the location, orientation

See Actuator Common Options for common options.

Special Options:

Constraint Mode
Menu specifying type of constraint required.

- Force Field Constraint
- Orientation Constraint
- Distance Constraint
- Location Constraint



Constraint actuator - Force Field

Force Field Constraint
Create a force field buffer zone along one axis of the object.

Damping
: Damping factor of the Fh spring force (Range 0.0 - 1.0)

Distance
: Height of Fh area

Rot Fh
: Make game object axis parallel to the normal of trigger object.

Direction
: Axis in which to create force field (can be + or -, or None)

Force

N
: When on, use a horizontal spring force on slopes

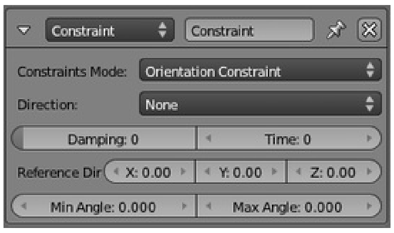
M/P
: Trigger on another Object will be either Material (M) or Property (P)

Property
: Property/Material that triggers the Force Field constraint (blank for ALL Properties/Materials)

Per
: Persistence button
: When on, force field constraint always looks at Property/Material; when off, turns itself off if it can't find the Property/Material.

Time
: Number of frames for which constraint remains active

RotDamp
: Damping factor for rotation



Constraint Actuator - Orientation

Orientation Constraint
Constrain the specified axis in the Game to a specified direction in the World axis.

Direction
: Game axis to be modified (X, Y, Z or none)

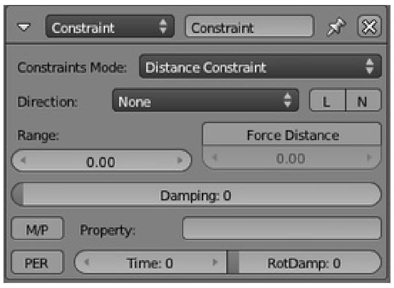
Damping
: Delay (frames) of the constraint response (0 - 100)

Time
: Time (frames) for the constraint to remain active (0 - 100)

ReferenceDir
: Reference direction (global coordinates) for the specified game axis.

MinAngle
: Minimum angle for the axis modification;

MaxAngle
: Maximum angle for the axis modification;



Constraint actuator - Distance

Distance Constraint
Maintain the distance the Game Object has to be from a surface

Direction
: Axis Direction (X, Y, Z, -X, -Y, -Z, or None)

L
: If on, use local axis (otherwise use World axis)

N
: If on, orient the Game Object axis with the mesh normal.

Range
: Maximum length of ray used to check for Material/Property on another game object (0 - 2000 Blender Units)

Force Distance
: •Distance to be maintained between object and the Material/Property that triggers the Distance Constraint(-2000 to +2000 Blender Units).

Damping
: Delay (frames) of the constraint response (0 - 100)

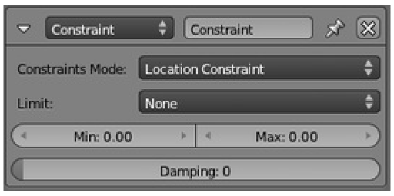
M/P
: Trigger on another Object will be either Material (M) or Property (P)

Property
: Property/Material that triggers the Force Field constraint (blank for ALL Properties/Materials)

Per
: Persistence button: When on, force field constraint always looks at Property/Material; when off, turns itself off if it can't find the Property/Material.

Time
: Number of frames for which constraint remains active

RotDamp
: Damping factor for rotation



Constraint actuator - Location

Location Constraint
Limit the position of the Game Object within one World Axis direction. To limit movement within an area or volume, use two or three constraints.

Limit
: Axis in which to apply limits (LocX, LocY, LocZ or none)

Min
: Minimum limit in specified axis (Blender Units)

Max
: Maximum limit in specified axis (Blender Units)

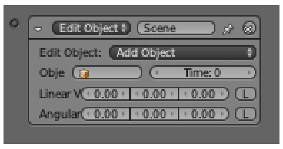
Damping
: Delay (frames) of the constraint response (0 - 100)

Edit Object Actuator

The Edit Object actuator allows the user to edit settings of objects in game

See Actuator Common Options for common options.

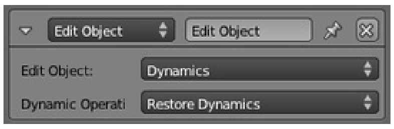
Special Options:



Edit Object actuator

Edit Object
Menu of options for Edit Object actuator

- Dynamics
- Track To
- Replace Mesh
- End Object
- Add Object



Edit Object actuator - Dynamics

Dynamics
Provides a menu of Dynamic Operations to set up dynamics options for object.

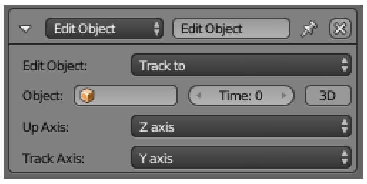
Set Mass
: Enables the user to set the mass of the current object for Physics (Range 0 - 10,000).

Disable Rigid Body
: Disables the Rigid Body state of the object - disables collision.

Enable Rigid Body
: Disables the Rigid Body state of the object - enables collision.

Suspend Dynamics
: Suspends the object dynamics (object velocity).

Restore Dynamics
: Resumes the object dynamics (object velocity).



Edit Object actuator - Track to

Track to
Makes the object “look at” another object, in 2D or 3D.

Object
: Object to follow.

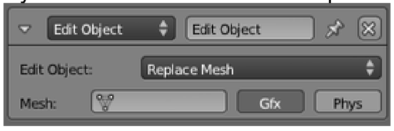
Time
: No. of frames it will take to turn towards the target object (Range 0-2000).

3D Button (toggle)
: Enable 2D (X,Y) or 3D (X,Y,Z) tracking.

Up Axis menu
: The axis (X,Y,Z) that points upward (Z by default).

Track Axis menu
: The axis (X,Y,Z,-X,-Y,-Z) that points to the target object (Y by default).

If you choose the same axis for Up Axis and Track Axis menus, the track to actuator will not be functional anymore.



Edit Object actuator - Replace Mesh

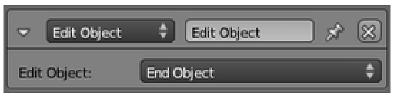
Replace Mesh
Replace mesh with another. Both the mesh and/or its physics can be replaced, together or independently.

Mesh

name of mesh to replace the current mesh.

Gfx Button
: replace visible mesh.

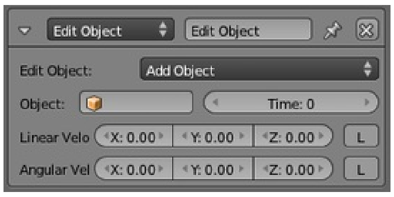
Phys Button
: replace physics mesh (not compound shapes)



Edit Object actuator - End Object

End Object

Destroy the current object (Note, debug properties will display error Zombie Object in console)



Edit Object actuator - Add Object

Add Object

Adds an object at the centre of the current object. The object that is added needs to be on another, hidden, layer.

Object
: The name of the object that is going to be added.:;Time: the time (in frames) the object stays alive before it disappears. Zero makes it stay forever.

Linear Velocity
: Linear Velocity, works like in the motion actuator but on the created object instead of the object itself. Useful for shooting objects, create them with an initial speed.

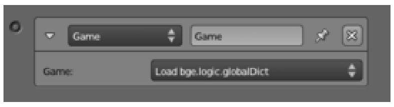
Angular Velocity
: Angular velocity, works like in the motion actuator but on the created object instead of the object itself.

Game Actuator

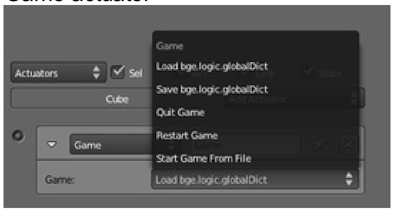
The Game actuator allows the user to perform Game-specific functions, such as Restart Game, Quit Game and Load Game.

See Actuator Common Options for common options.

Special Options:



Game actuator


Actuators
Sel
Cube
Game
Game:
Game
Load bge.logic.globalDict
Save bge.logic.globalDict
Quit Game
Restart Game
Start Game From File
Load bge.logic.globalDict
Game

Game

Load bge.logic.globalDict
: Load bge.logic.globalDict from .bgeconf.

Save bge.logic.globalDict
: Save bge.logic.globalDict to .bgeconf.

Quit Game
: Once the actuator is activated, the blenderplayer exits the runtime.

Restart Game
: Once the actuator is activated, the blenderplayer restarts the game (reloads from file).

Start Game From File
: Once the actuator is activated, the blenderplayer starts the .blend file from the path specified.

  File

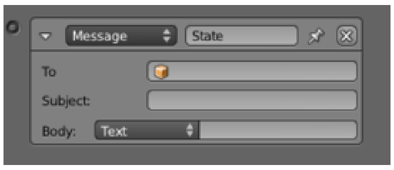
  : Path to the .blend file to load.

Notes

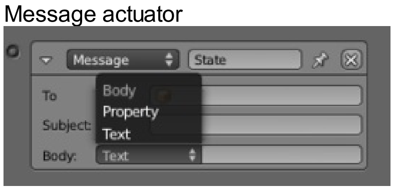
If you use the keyboard sensor as a hook for the Esc key, in the event that the quit game actuator fails, such as an error in a python file, the game will be unable to close. Data may be recovered from quit.blend File » Recover Last Session

Message Actuator

The Message actuator allows the user to send data across a scene, and between scenes themselves.



Message actuator



Message actuator Options

See Actuator Common Options for common options.

Special Options:

To
: Object to broadcast to. Leave blank if broadcast to all (or sending to another scene).

Subject
: Subject of message. Useful if sending certain types of message, such as "end-game", to a message sensor listening for "end game"->AND->Quit Game actuator

Body
: Body of message sent (only read by Python*).

Text

User specified text in body.

Property

User specified property.

Usage Notes

You can use the Message Actuator to send data, such as scores to other objects, or even across scenes! (alternatively use bge.logic.globalDict).

Motion Actuator

The Motion actuator sets an object into motion and/or rotation. There are two modes of operation, simple or servo in which the object can either teleport, rotate or dynamically move. Also, simple mode operation depends on the type of Physics setting for the Object.

See Actuator Common Options for common options.

Special Options:

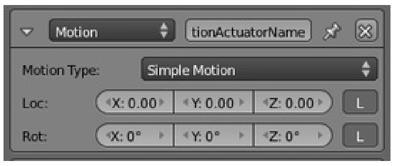
Motion Type
Determines type of motion

Simple Motion
applies different kinds of motions directly
Servo Control
sets a target speed and also how quickly it reaches that speed.

Object collisions

Simple motion can cause an object to go through another object since it never passes the any of the coordinates between the start and end. This can be avoided using Servo Control, which is activated when the Physics setting for the object(s) is set to Dynamic/Rigid Body/Soft Body.

Simple Motion


Motion
tionActuatorName
Motion Type: Simple Motion
Loc: X: 0.00 Y: 0.00 Z: 0.00 L
Rot: X: 0° Y: 0° Z: 0° L

Motion actuator for Simple Motion

Loc
The object jumps the number of blender units entered, in each of the three axes,each time a pulse is received.

Rot
The object rotates by the specified amount, in each of the three axes, each time a pulse is received.

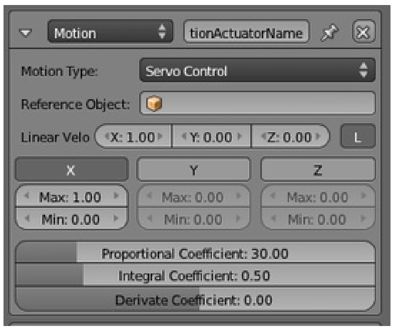
The rotation is specified in degrees. If you have enabled tooltips in the user preferences, the tip will also show the value converted in radians.

L
Coordinates specified are Global (Black and gray) or Local (White and gray).

Servo Control

To make Servo Control work, it is necessary to turn on Dynamic in the Physics window, and to make the object an Actor.

Servo Control


Motion
tionActuatorName
Motion Type: Servo Control
Reference Object:
Linear Velo X: 1.00 Y: 0.00 Z: 0.00 L
X Y Z
Max: 1.00 Max: 0.00 Max: 0.00
Min: 0.00 Min: 0.00 Min: 0.00
Proportional Coefficient: 30.00
Integral Coefficient: 0.50
Derivate Coefficient: 0.00

Motion actuator for Servo Control

Servo control is a powerful way to achieve motion in way which mimics the movement of objects in the physical world. It consists in a servo controller that adjusts the force on the object in order to achieve a given speed. Uses the Proportional - Integral - Derivative (PID) equations of motion See Ref..

Reference Ob
Specifies the object which the actuator owner uses as a reference for movement, for moving platforms for example. If empty it will use world reference.

Linear V
The target linear velocity, in each of the three axes, which the object will try and achieve.

L
Coordinates specified are Global (gray) or Local (White).

X, Y, Z
: Sets maximum and minimum limits for the force applied to the object. If disabled (i.e. X,Y or Z buttons are gray) the force applied is unlimited.

Proportional Coefficient
: Set the Proportional Coefficient. This controls the reaction to differences between the actual and target linear velocity.

Integral Coefficient
: Set the Integral Coefficient. This controls the reaction to the sum of errors so far in this move.

Derivative Coefficient
: Set the Derivative Coefficient. This controls the reaction

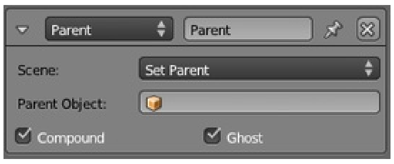
Parent Actuator

Enables you to change the parent relationships of the current object.

See Actuator Common Options for common options.

Special Options:

Scene
Menu for parenting operation required.



Parent Actuator

Set Parent
Make this object to be current object's parent

Parent Object

Name of parent object

Compound'

Add this object shape to the parent shape (only if the parent shape is already compound)

Ghost'

Make this object ghost while parented

Remove Parent
Remove all parents of current object

Parent Object

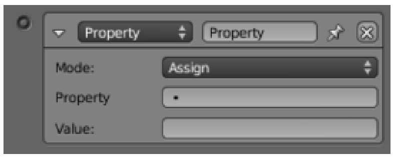
Name of parent object

Property Actuator

Using the Property actuator you can change the value of a given property once the actuator itself is activated.

See Actuator Common Options for common options.

Special Options:



Property actuator

Mode

Assign
: the Property target property will become equal to the set Value once the actuator is activated

Add
: adds Value to the value of the property Property once the actuator is activated (enter a negative value to decrease). For Bool, a value other than 0 (also negative) is counted as True.

Copy
: copies a property from another object to a property of the actuator owner once the actuator is activated.

Toggle
: switches 0 to 1 and any other number than 0 to 0 once the actuator is activated. Useful for on/off switches.

Level
: switches to 1 if the actuator is activated and back to 0 if the actuator is deactivated.

Property
: The target property that this actuator will change

Value
: The value to be used to change the property

Example

You have a character, it has a property called “hp” (hit points) to determine when he has taken enough damage to die. hp is an int with the start value of 100.

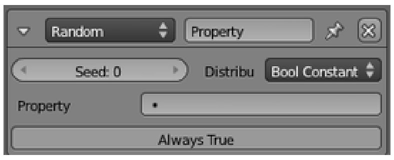
You set up two Collision sensors, one for enemy bullets, and one for picking up more health. The first one is connected (through an AND controller) to an Add Property actuator with the property hp and the value -10. Every time the player is hit by an enemy bullet he loses 10 hp. The other sensor is connected (through an AND controller) to an other Add Property actuator, this one with the value 50. So every time the player collides with a health item the hp increases by 50. Next you set up a Property sensor for an interval, greater than 100. This is connected (through an AND controller) to an Assign Property actuator which is set to 100. So if the players hp increases over 100 it is set to 100.

Random Actuator

Sets a random value into a property of the object

See Actuator Common Options for common options.

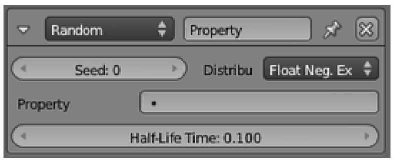
Special Options:



Camera Actuator

Seed
Starting seed for random generator (range 1 - 1000)

Distribution
Menu of distributions from which to select the random value. The default entry of Boolean Constant gives either True or False, which is useful for test purposes.



Float Neg. Exp.

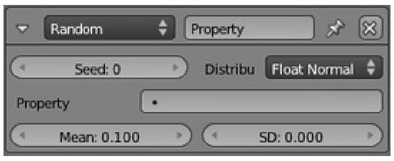
Float Neg. Exp.
Values drop off exponentially with the specified half-life time.

Property

Float property to receive value

Half-Life Time

Half-life time (Range 0.00 -10000.00)



Float Normal

Float normal
Random numbers from a normal distribution.

Property

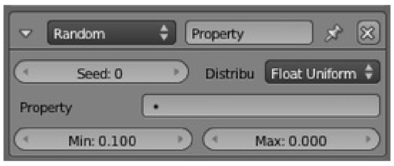
Float property to receive value

Mean

Mean of normal distribution (Range -10000.00 to +10000.00)

SD

Standard deviation of normal distribution (Range 0.00 to +10000.00)



Float Uniform

Float uniform
Random values selected uniformly between maximum and minimum.

Property

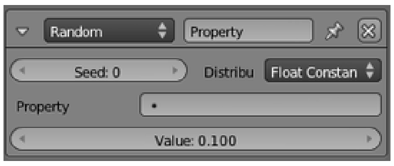
Float property to receive value

Min

Minimum value (Range -10000.00 to +10000.00)

Max

Maximum value (Range -10000.00 to +10000.00)



Float Constant

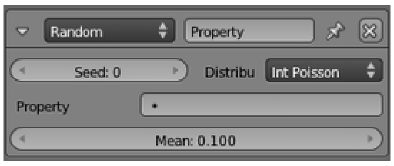
Float constant
Returns a constant value.

Property

Float property to receive value

Value

Value (Range 0.00 to +1.00)



Random Integer Poisson

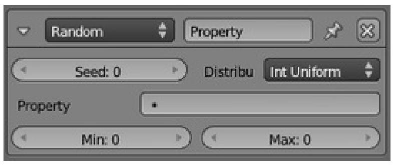
Int Poisson
Random numbers from a Poisson distribution.

Property

Integer property to receive value

Mean

Mean of Poisson distribution (Range 0.01 to +100.00)



Random Integer Uniform

Int uniform
Random values selected uniformly between maximum and minimum.

Property

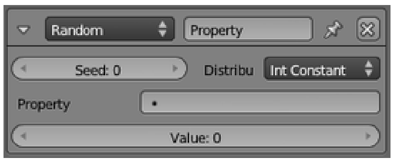
Integer property to receive value

Min

Minimum value (Range -1000 to +1000)

Max

Maximum value (Range -1000 to +1000)



Random Integer Constant

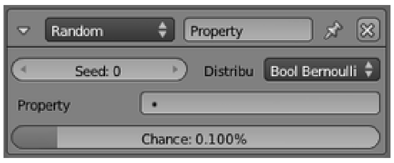
Int constant
Returns a constant value.

Property

Integer property to receive value

Value

Value (Range 0.00 to +1.00)



Random Bool Bernoulli

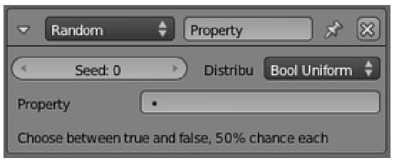
Bool Bernoulli
Returns a random distribution with specified ratio of TRUE pulses.

Property

Boolean property to receive value

Chance

Proportion of TRUE responses required.

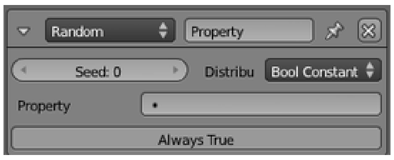


Random Bool Uniform

Bool uniform
A 50/50 chance of obtaining True/False.

Property

Boolean property to receive value



Random Bool Constant

Bool constant
Returns a constant value.

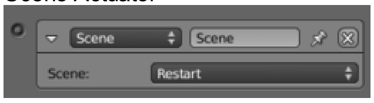
Property

Boolean property to receive value

Value

Value (True or False)

Scene Actuator

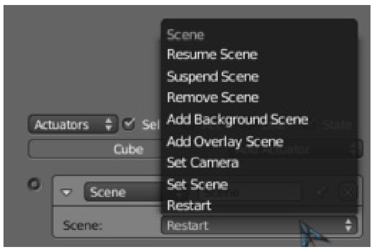


Scene actuator

The Scene actuator manages the scenes in your .blend file, these can be used as levels or for UI and background.

See Actuator Common Options for common options.

Special Options: The actuator has eight modes:



Scene actuator options

Restart
: Restarts the current scene, everything in the scene is reset

Set Scene
: Changes scene to selected one

Set Camera
: Changes which camera is used

Add OverlayScene
: This adds an other scene, and draws it on top of the current scene. It is good for interfacing: keeping the health bar, ammo meter, speed meter in an overlay scene makes them always visible.

Add BackgroundScene
: This is the opposite of an overlay scene, it is drawn behind the current scene

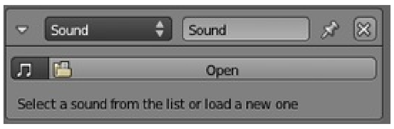
Remove Scene
: Removes a scene.

Suspend Scene
: Pauses a scene

Resume Scene
: Resumes a paused scene.

Sound Actuator

Select a sound file from the list or make a new one.



Select a sound from the list or load a new one

Sound Actuator

See Actuator Common Options for common options.

Special Options:

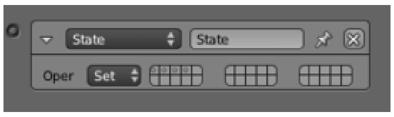
Music File title
: Select music file from the list presented.

State Actuator

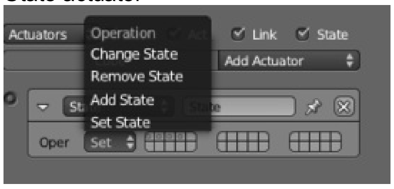
The State actuator allows the user to create complex logic, whilst retaining a clear user interface. It does this by having different states, and performing operations upon them

See Actuator Common Options for common options.

Special Options:



State actuator



State actuator options

Operation

Menu to select the state operation required.

Change State
Change from the current state to the state specified.
Remove State
Removes the specified states from the active states (deactivates them).
Add State
Adds the specified states to the active states (activates them).
Set State
Moves from the current state to the state specified, deactivating other added states.

Usage Notes

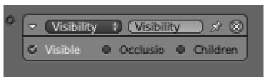
With the state actuator, you can create tiers of logic, without the need for hundreds of properties. Use it well, and you benefit greatly, but often problems may be circumvented by python.

Steering Actuator

Under Construction (7 July 2012)

Visibility Actuator

The Visibility actuator allows the user to change the visibility of objects during runtime.



Visibility actuator

See Actuator Common Options for common options.

Special Options:

Visible
: Toggle checkbox to toggle visibility

Occlusion
: Toggle checkbox to toggle occlusion. Must be initialised from the Physics tab.

Children
: Toggle checkbox to toggle recursive setting - will set visibility / occlusion state to all child objects, children of children (recursively)

Usage Notes

Using the visiblity actuator will save on Rasterizer usage, however not Physics, and so is limited in terms of Level of Detail (LOD). For LOD look at replace mesh, but be aware that the logic required can negate the effect of the LOD.

Properties

Properties are the game logic equivalent to variables. They are stored with the object, and can be used to represent things about them such as ammo, health, name, and so on.

Property Types

There are five types of properties:

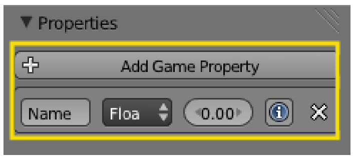
Timer	Starts at the property value and counts upwards as long as the object exists. It can for example be used if you want to know how long time it takes the player to complete a level.
Float	Uses decimal numbers as values, can range from -10000.000 to 10000.000. It is useful for precision values.
Integer	Uses integers (whole numbers) as values, between -10000 and 10000. Useful for counting things such as ammunition, where decimals are unnecessary.
String	Takes text as value. Can store 128 characters.
Boolean	Boolean variable, has two values: true or false. This is useful for things that have only two modes, like a light switch.

Using Properties

Properties can be set up and initialised in the Properties panel of the Logic Editor - see the Property Editing page for details. When a game is running, values of properties are set, manipulated, and evaluated using the Property Sensor and the Property Actuator.

Property Editing

Logic Properties are created and edited using the panel on the left of the Logic Editor Panel. The top menu provides a list of the available property types.



Property Panel

Add Property button
: This button adds a new property to the list, default is a Float property named “`prop`”, followed by a number if there already is one with this name.

Name field
: Where you give your property its name, this is how you are going to access it through python or expressions. The way to do so in python is by dictionary style lookup (`GameObject["`*`propname`*`"]`). The name is case sensitive.

Type menu
: This menu determines which type of property it is (see below).

Value field
: Sets the initial value of the property.

I information button
: Display property value in debug information. If debugging is turned on, the value of the property is given in the top left-hand corner of the screen while the game is run. To turn debugging on, tick Show Debug Properties in the Game menu. All properties with debugging activated will then be presented with their object name, property name and value during gameplay. This is useful if you suspect something with your properties is causing problems.

X
: Delete property.

States

In the BGE, an object can have different "states". At any time while the game is playing, the current state of the object defines its behavior. For instance, a character in your game may have states representing awake, sleeping or dead. At any moment their behaviour in response to a loud bang will be dependant on their current state; they may crouch down (awake); wake up (asleep) or do nothing (dead).

How States Operate

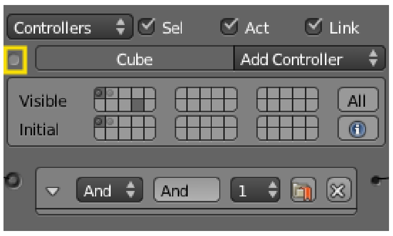
States are set up and used through controllers: note that only controllers, not actuators and sensors, are directly controlled by the state system. Each object has a number of states (up to 30; default = 1), and can only be in one state at any particular time. A controller must always specify the state for which it will operate - it will only give an output pulse if a) its logic conditions are met, and b) the object is currently in the specified State. States are set up and edited in the object's Controller settings (for details see below).

State settings are automatic in simple games. By default, the number of states for each object is 1, and all controllers are set to use State 1. So, if a game does not need multiple states, everything will work without explicitly setting states - you do not need to bother about states at all.

{{{2}}}

One of the actuators, the State actuator, can set or unset the object's State bits, and so allow the object's reaction to a sensor signal to depend on its current state. So, in the above example, the actor will have a number of controllers connected to the “loud bang” sensor, for each of the “awake”, “asleep” or “dead” states. These will operate different actuators depending on the current state of the actor, and some of these actuators may switch the actor's state under appropriate conditions.

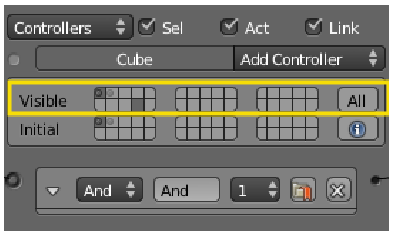
Editing States



State Panel Button

States are set up and edited using the Controller (center) column of the Game Logic Panel. To see the State panel, click on the State Panel Button shown. The panel shows two areas for each of the 30 available states; these show Visible states, and Initial states (see below|). Setting up the State system for a game is performed by choosing the appropriate state for each controller in the object's logic.

The display of an object's state logic, and other housekeeping, is carried out using the State Panel for the object, which is switched on and off using the button shown. The panel is divided into two halves, Visible and Initial.

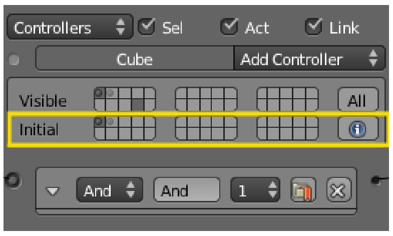


State Panel Visible

Visible States

In the Visible area, each of the 30 available states is represented by a light-gray square. This panel shows what logic is visible for the logic brick displayed for the object. At the right is the All button; if clicked, then all the object's logic bricks are displayed (this is a toggle), and all State Panel squares are light-gray. Otherwise, individual states can be clicked to make their logic visible. (Note that you can click more than one square). Clicking the square again unselects the state.

States for the object that are in use (i.e. the object has controllers which operate in that state) have dots in them, and squares are dark-gray if these controllers are shown in the Game Logic display. The display of their connected sensors and actuators can also be controlled if the State buttons at the head of their columns are ticked.



State Panel Initial

Initial State

In the Initial area, each of the 30 available states is again represented by a light-gray square. One of these states may be clicked as the state in which the object starts when the game is run.

At the right is the I (Information) button; if clicked, and the (Game) Show Debug Properties menu entry is clicked, the current state of the object is shown in the top left-hand corner of the display while the game is running.

Camera

The Game Engine camera is in many ways similar to the Camera in the normal Blender Render system, and is created, parameterized and manipulated in similar ways. However because of its use as a real-time device, the Game Engine camera has a number of additional features - it may be used as not only as a static camera, but also as a moving device with its default characteristics (ie. with its own programmed moves), or it may track another object in the game. Furthermore, any game object may be used as a camera; the view is taken from the object's origin point. Lastly, it may be given special capabilities such as Stereo vision, Dome visualisation etc. which have special relevance to game technology.

When you start the Game Engine, the initial camera view is taken from the latest 3D View. This may be either a selected camera object or the default camera (see below). Thus to start the game with a particular camera, you must select the camera and press 0 NumPad before starting the Game Engine.

To avoid camera distortion

Always zoom the view in until the camera object fills the entire viewport.

Default Camera

The default camera view is taken from the latest 3D viewport view, at a distance equivalent to the viewer. This means that if the normal 3D view is active the scene does not change when the Game Engine is started.

Camera Object

The Camera object in the Game Engine follows much the same structure as the conventional Blender camera - see Camera for details of how to set up, manipulate and select a camera. The following sections show some of the special facilities available in BGE cameras.

Parent Camera to Object

The camera will follow the object. First select the camera and then select the object. Next CtrlP → Make Parent.

Note that if your object has any rotations then the camera will also have those rotations. To avoid this use “Parent to Vertex” (see below).

Parent to Vertex

The easiest way to accomplish this is to select your object and ⇆ Tab to Edit mode. Now select the vertex and ⇆ Tab back to Object mode.

Next, without any objects selected, select the camera and, holding the ⇧ Shift key, select the object. ⇆ Tab into Edit mode, and CtrlP and choose Make vertex parent.

Now the camera will follow the object and it will maintain its rotation, while the object rotates.

Object as a Camera

Any object may also become a camera with whatever properties are set for the object.

To make an object the camera, in Object mode select the object and press Ctrl0 NumPad on the numpad.

To reverse it, just select the camera and Ctrl0 NumPad again.

Camera Lens Shift

In the Blender interface, there is an option to shift the camera view on the x/y plane of the view. It is comparable to lens shift in video projectors that usually shift the image up along the Y axis. So for example, when you put the beamer on a table it does not project half of the image on the table.

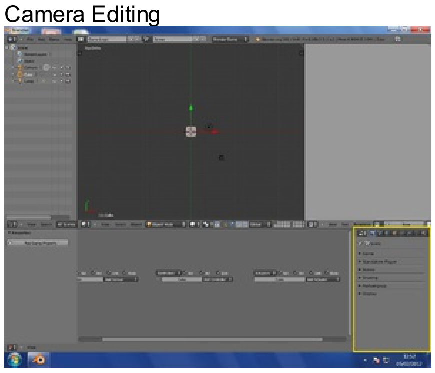
Unfortunately, this parameter is not taken in account by the Game Engine.

To manipulate the projection we can then directly modify the camera projection matrix in Python.

```
import bge
scene = bge.logic.getCurrentScene()
cam = scene.active_camera
# get projection matrix
camatrix = cam.projection_matrix
#modifying the camera projection matrix by modifying the x and y terms of the 3rd row to obtain a shift of the rendered area
camatrix[2][0] = 2*shiftx
camatrix[2][1] = 2*shitfy
cam.projection_matrix = camatrix
```

Here in field of view units are shiftx and shifty. So for example, for shifting the view up half a screen shifty is set to 0.5.

Note that a camera's projection_matrix attribute may not be set until after initialization scripts are executed and running this code immediately after the game starts will mess up the projection matrix.

Camera Editing

Camera Properties

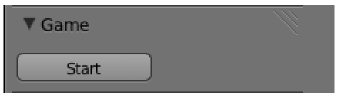
The camera (or cameras) used in a Blender game have a wide-ranging effect on the way in which the game is rendered and displayed. Mostly this is controlled using the Properties panel of the camera(s) used in the game.

Render Engine

Make sure that the render engine is set to Blender Game when attempting to set these controls - otherwise this description will not tally with what you see!

In the Camera Properties area, there are six panels available, as shown. Each can be expanded or contracted using the usual triangle button. The features in each panel will be described in detail below.

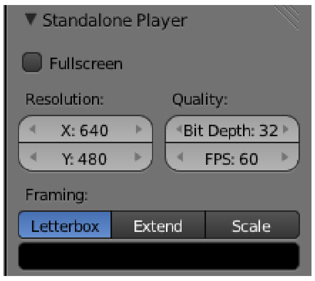
Embedded Player



Game Panel

Start button - Start the Game Engine. Shortcut P.

Standalone Player



Standalone Panel

This panel provides information for the Standalone Game Player which allows games to be run without Blender. See Standalone Player for further details.

Fullscreen -
Off - opens standalone game as a new window.
On - opens standalone game in full screen.

Resolution

X

Sets the X size of the viewport for full-screen display.

Y

Sets the Y size of the viewport for full-screen display.

Quality

Bit Depth

Number of bits used to represent color of each pixel in full-screen display.

FPS

Number of frames per second of full-screen display.

Framing

Shows how the display is to be fitted in to the viewport.

Letterbox

Show the entire viewport in the display window, and fill the remainder with the "bar" color.

Extend

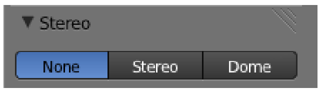
Show the whole display in the viewport, and fill the remainder with bars.

Scale

Scale the display in X and Y to exactly fill the entire viewport.

Bar Color
Select a color to use as the color of bars around the viewport (default black).
To use this, select a color mode (RGB, HSV or Hex), then use the color slider and color wheel to choose a bar color.

Stereo



Stereo Panel

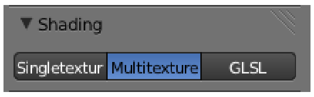
Select a stereo mode that will be used to capture stereo images of the game (and also, by implication, that stereo displays will use to render images in the standalone player).

None
Render single images with no stereo.

Stereo
Render dual images for stereo viewing using appropriate equipment. See Stereo Camera for full details of available options.

Dome
Provides facilities for an immersive dome environment in which to view the game. See Dome Camera for full details of available options.

Shading



Shading Panel

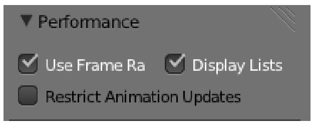
Specifies the shading mode to be used in rendering the game.The shading facilities available in Blender for use in Materials and Textures are essentially the same in the Blender Game Engine. However the constraints of real-time display mean that only some of the facilities are available.

Single Texture
Use single texture facilities.

Multitexture
Use Multitexture shading.

GLSL
Use GLSL shading. GLSL should be used whenever possible for real-time image rendering.

Performance



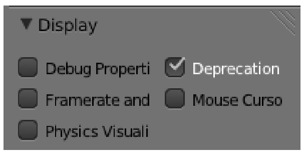
Performance Panel

Use Frame Rate
Respect the frame rate rather than rendering as many frames as possible.

Display Lists
Use display lists to speed up rendering by keeping geometry on the GPU.

Restrict Animation Updates
Restrict number of animation updates to the animation FPS (this is better for performance but can cause issues with smooth playback).

Display



Display Panel

Gives various display options when running the Game Engine. under the .

Debug Properties
Show properties marked for debugging while game runs. Note that debug properties to be shown must be requested at source (eg. i-button in state tables). Only available when game is run within Blender - not in standalone player version.

Framerate and Profile
Show framerate and profiling information while game runs. Only available when game is run within Blender - not in standalone player version.

Physics Visualization
Show physics bounds and interactions while game runs (available in both Blender and standalone versions).

Deprecation Warnings
Print warnings when using deprecated features in the python API. Only available when game is run within Blender - not in standalone player version.

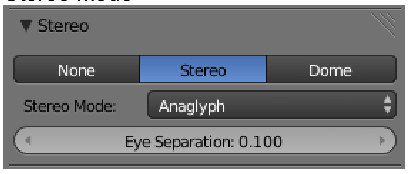
Mouse Cursor
Show mouse cursor while game runs (available in both Blender and standalone versions).

Stereo Camera

Stereo Cameras allow you to generate images that appear three dimensional when wearing special glasses. This is achieved by rendering two separate images from cameras that are a small distance apart from each other, simulating how our own eyes see. When viewing a stereo image, one eye is limited to seeing one of the images, and the other eye sees the second image. Our brain is able to merge these together, making it appear that we are looking at a 3d object rather than a flat image. See Stereoscopy for more information on different stereoscopic viewing methods.

Stereo Settings

Stereo Mode



Set the type of stereo camera to use. Possible modes are detailed below.

Eye Separation

This value is extremely important. It determines how far apart the two image-capturing cameras are, and thus how "deep" the scene appears. Too small a value and the image appears flat; too high a value can result in headaches and eye strain. The ideal value mimics the separation of the viewer's two eyes.

Stereo Modes

Specifies the way in which the left-eye image and the right-eye image pixels are put together during rendering. This must be selected according to the type of apparatus available to display the appropriate images to the viewer's eyes.

Anaglyph

One frame is displayed with both images color encoded with red-blue filters. This mode only requires glasses with color filters, there are no special requirements for the display screen and GPU.

Quad Buffer

Uses double buffering with a buffer for each eye, totaling four buffers (Left Front, Left Back, Right Front and Right Back), allowing to swap the buffers for both eyes in sync. See Quad Buffering for more information.

Side by Side

Lines are displayed one after the other, so providing the two images in two frames side by side.

Above-Below

Frames are displayed one after the other, so providing the two images in two frames, one above the other.

Interlaced

One frame is displayed with the two images on alternate lines of the display.

Vinterlaced

One frame is displayed with both images displayed on alternate columns of the display. This works with some 'autostereo displays'.

3D Tv Top-Bottom

One frame displays the left image above and the right image below. The images are squashed vertically to fit. This mode is designed for passive 3D TV.

Dome Camera

This feature allows artists to visualize their interactive projects within an immersive dome environment. In order to make it an extensible tool, we are supporting Fulldome, Truncated domes (front and rear), Planetariums and domes with spherical mirrors.

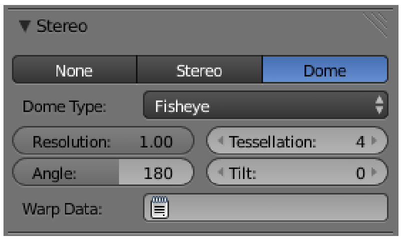
The Dome camera uses a multipass texture algorithm as developed by Paul Bourke and was implemented by Dalai Felinto with sponsorship from **SAT -** Society for Arts and Technology within the **SAT Metalab** immersion research program.[1]Briefly, that involves rendering the scene 4 times and placing the subsequent images onto a mesh designed especially such that the result, when viewed with an orthographic camera, is a fisheye projection.

Note

Remember to use Blender in **fullscreen mode** to get the maximum out of your projector.

To accomplish that launch Blender with the command-line argument -W. Also to get away of the top menu on Blender try to join all windows (buttons, 3dview, text, ...) in a single one. Otherwise if you only maximize it (Ctrl+Up) you can't get the whole screen free to run your game (the top bar menu takes about 20 pixels).

Dome Camera Settings


▼ Stereo
None | Stereo | Dome
Dome Type: Fisheye
Resolution: 1.00 | Tessellation: 4
Angle: 180 | Tilt: 0
Warp Data:

Dome Type
: This menu allows you to select which type of dome camera to use. They are outlined below, along with their respective settings.

- Fisheye Dome
- Front-Truncated Dome
- Rear-Truncated Dome
- Cube Map
- Full Spherical Panoramic

Available camera settings change depending on the selected Dome Type:

Resolution
: Sets the resolution of the Buffer. Decreasing this value increases speed, but decreases quality.

Tesselation
: 4 is the default. This is the tesselation level of the mesh. (Not available in Cube Map mode).

Angle
: Sets the field of view of the dome in degrees, from 90 to 250. (Available in Fisheye and Truncated modes).

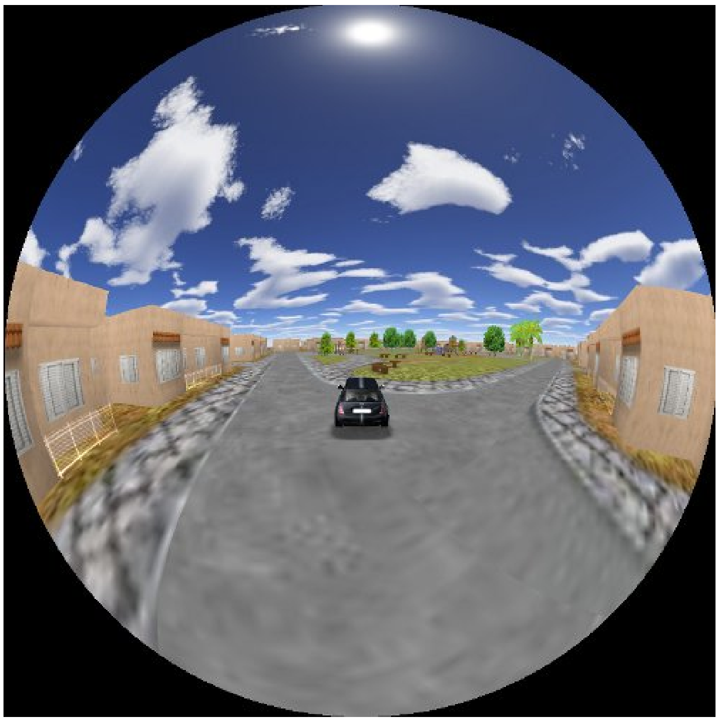
Tilt
: Set the camera rotation in the horizontal axis. Available in Fisheye and Truncated modes).

Warp Data
: Use a custom warp mesh data file.

Fisheye Mode

An Orthogonal Fisheye view from 90° to 250° degrees.

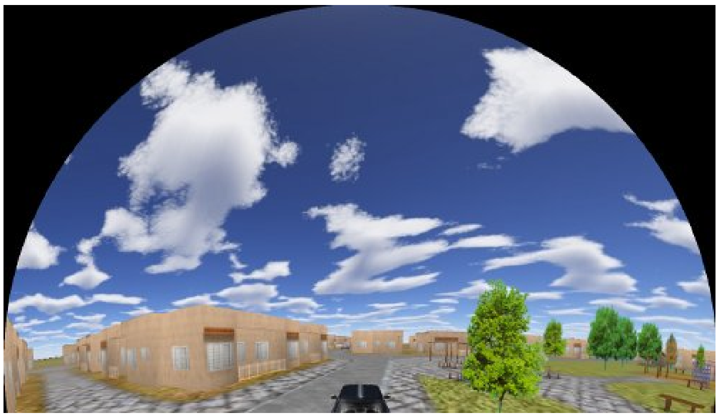
- From 90° to 180° we are using 4 renders.
- From 181° to 250° we are using 5 renders.



Front-Truncated Dome Mode

Designed for truncated domes, this mode aligns the fisheye image with the top of the window while touching the sides.

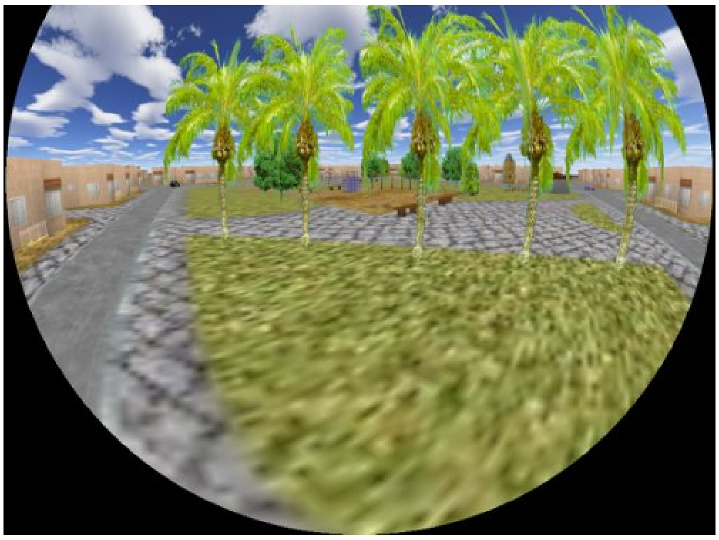
- The Field of view goes from 90º to 250º degrees.
- From 90º to 180º we are using 4 renders.
- From 181º to 250º we are using 5 renders.



Rear-Truncated Dome Mode

Designed for truncated domes, this mode aligns the fisheye image with the bottom of the window while touching the sides.

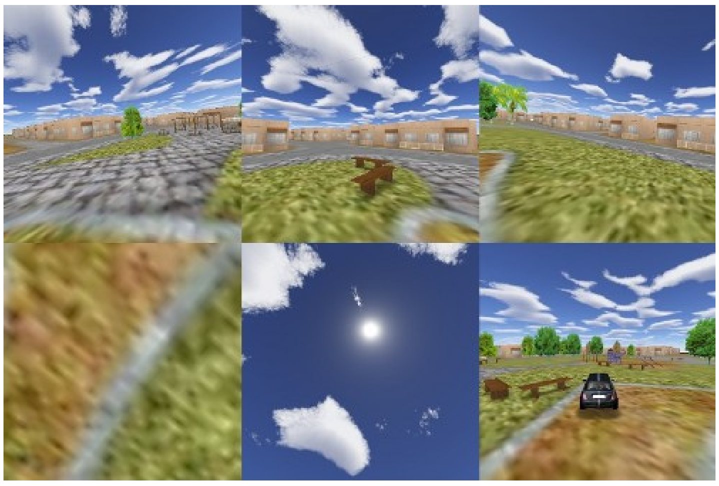
- The Field of view goes from 90º to 250º degrees.
- From 90º to 180º we are using 4 renders.
- From 181º to 250º we are using 5 renders.



Cube Map Mode

Cube Map mode can be used for pre-generate animated images for CubeMaps.

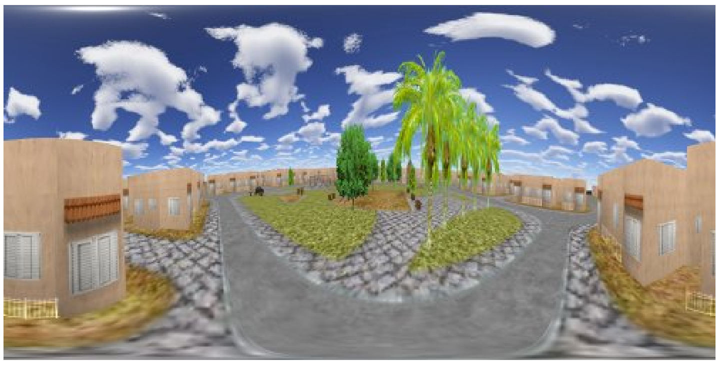
- We are using 6 renders for that. The order of the images follows Blender internal EnvMap file format:
  - first line: right, back, left
  - second line: bottom, top, front



Spherical Panoramic

A full spherical panoramic mode.

- We are using 6 cameras here.
- The bottom and top start to get precision with **Definition** set to 5 or more.



Warp Data Mesh

Many projection environments require images that are not simple perspective projections that are the norm for flat screen displays. Examples include geometry correction for cylindrical displays and some new methods of projecting into planetarium domes or upright domes intended for VR.

For more information on the mesh format see Paul Bourke's article.

In order to produce that images, we are using a specific file format.

File template::

```
mode
width height
n0_x n0_y n0_u n0_v n0_i
n1_x n1_y n1_u n1_v n1_i
n2_x n1_y n2_u n2_v n2_i
n3_x n3_y n3_u n3_v n3_i
(...)
```

First line is the image type the mesh is support to be applied to: **2** = **rectangular**, **1** = **radial**Next line has the mesh dimensions in pixelsRest of the lines are the nodes of the mesh.

Each line is compund of **x y u v i**(x,y) are the normalised screen coordinates(u,v) texture coordinatesi a multiplicative intensity factor

x varies from -screen aspect to screen aspecty varies from -1 to 1u and v vary from 0 to 1i ranges from 0 to 1, if negative don't draw that mesh node

- You need to create the file and add it to the Text Editor in order to select it as your Warp Mesh data file.
- Open the Text Editor (Window Types/Text Editor).
- Open your mesh data file(ie. myDome.data) in the text editor (Text/Open or Alt O on keyboard).
- Go to Game Framing Settings (Window Types/Buttons Window/Scene Page or F10 on keyboard)
- Enable Dome Mode.
- Type filename in Warp Data field(ie. myDome.data).

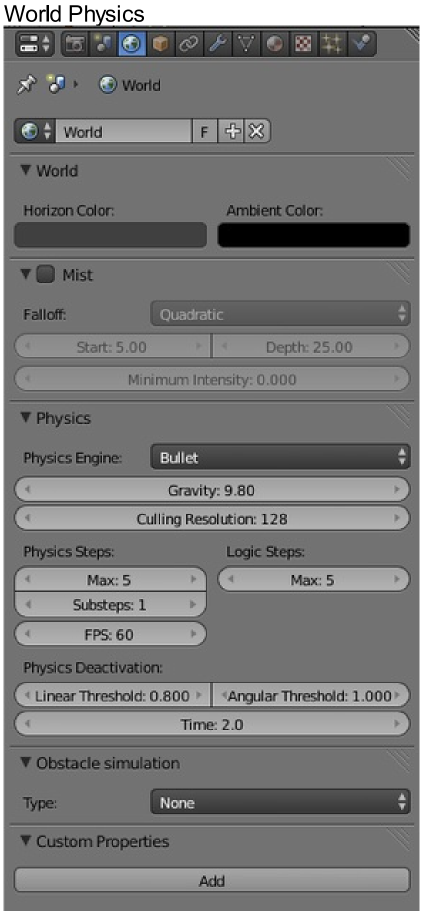
To create your own Warp Meshes an interactive tool called meshmapper is available as part of Paul Bourke's Warpplayer software package(requires full version).

Example files

Spherical Mirror Dome 4x3, Truncated Dome 4x3, Sample Fullscreen File 4x3, Sample Fullbuffer File 4x3.

Note
Important: the viewport is calculated using the ratio of canvas width by canvas height. Therefore different screen sizes will require different warp mesh files. Also in order to get the correct ratio of your projector you need to use Blender in Fullscreen mode.

World Physics



BGE World Panel (fully expanded)

World settings enable you to set some basic effects which affect all scenes throughout your game, so giving it a feeling of unity and continuity. These include ambient light, depth effects (mist) and global physics settings. These effects are a limited subset of the more extensive range of effects available with the Blender renderer (see Doc:2.6/Manual/World|World)

While world settings offer a simple way of adding effects to a scene, compositing nodes are often preferred, though more complex to master, for the additional control and options they offer. For example, filtering the Z value (distance from camera) or normals (direction of surfaces) through compositing nodes can further increase the depth and spacial clarity of a scene.

{{{2}}}

World

These two color settings allow you to set some general lighting effects for your game.

Horizon Color
: The RGB color at the horizon ; i.e. the color and intensity of any areas in the scene which are not filled explicitly.

Ambient Color
: Ambient light mimics an overall background illumination obtained from diffusing surfaces (see Ambient Light, Exposure and Ambient Occlusion). Its general color and intensity are set by these controls.

Mist

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender makes objects farther away more transparent (decreasing their Alpha value) so that they mix more of the background color with the object color. With Mist enabled, the further the object is away from the camera the less it's alpha value will be. For full details, see Mist.

Mist
: Toggles mist on and off

Falloff
: Sets the shape of the falloff of the mist.

Start
: The starting distance of the mist effect. No misting will take place for objects closer than this distance.

Depth
: The depth at which the opacity of objects falls to zero.

Minimum intensity
: Overall minimum intensity of the mist

Game Physics

The Game Physics located in the World panel determine the type of physical rules that govern the game engine scene, and the gravity value to be used. Based on the physics engine selected, in physics simulations in the game engine, Blender will automatically move Actors in the downward (-Z) direction. After you arrange the actors and they move as you wish, you can then bake this computed motion into fixed Ipo curves (see Logic actors for more info).

Physics Engine
: Set the type of physics engine to use.

  Bullet
  : The default physics engine, in active development. It handles movement and collision detection. The things that collide transfer momentum to the collided object.

  None
  : No physics in use. Things are not affected by gravity and can fly about in a virtual space. Objects in motion stay in that motion.

Gravity

The gravitational acceleration, in units of meters per squared second ($m.s^{-2}$), of this world. Each object that is an actor has a mass and size slider (see Object Physics section). In conjunction with the frame rate (see Render section), Blender uses this info to calculate how fast the object should accelerate downward.

Culling Resolution

The size of the occlusion culling buffer in pixel, use higher value for better precision (slower). The optimized Bullet DBVT for view frustum and occlusion culling is activated internally by default.

Physics Steps
Max

Sets the maximum number of physics steps per game frame if graphics slow down the game. higher value allows physics to keep up with realtime.

Substeps

Sets the number of simulation substeps per physics timestep. Higher value give better physics precision.

FPS

Set the nominal number of game frames per second. Physics fixed timestep = 1/fps, independently of actual frame rate.

Logic Steps

Sets the maximum number of logic frame per game frame if graphics slows down the game, higher value allows better synchronization with physics.

Physics Deactivation
: These settings control the threshold at which physics is deactivated. These settings help reducing the processing spent on Physics simulation during the game.

Linear Threshold

The speed limit under which a rigid bodies will go to sleep (stop moving) if it stays below the limits for a time equal or longer than the deactivation time (sleeping is disabled when deactivation time is set to 0).

Angular Threshold

Same as linear threshold, but for rotation limit (in rad/s)

Time

The amount of time in which the object must have motion below the thresholds for physics to be disabled (0.0 disables physics deactivation).

Obstacle Simulation

Simulation used for obstacle avoidance in the Game Engine, based on the RVO (Reciprocal Velocity Obstacles) principle. The aim is to prevent one or more actors colliding with obstacles. See Path finding and steering behaviors for more details.

Type
None

obstacle simulation is disabled, actors aren't able to avoid obstacles

RVO (cells)
: obstacle simulation is based on the RVO method with cell sampling.

RVO (rays)
: obstacle simulation is based on the RVO method with ray sampling

Level height
: Max difference in heights of obstacles to enable their interaction. Used to define minimum margin between obstacles by height, when they are treated as those which are situated one above the other i.e. they doesn't influence to each other.

Visualization
: Enable debug visualization for obstacle simulation.

Blender Game Physics

Blender includes advanced physics simulation in the form of the Bullet Physics Engine (BulletPhysics.org). Most of your work will involve setting the right properties on the objects in your scene---then you can sit back and let the engine take over. The physics simulation can be used for Games, but also for Animation.

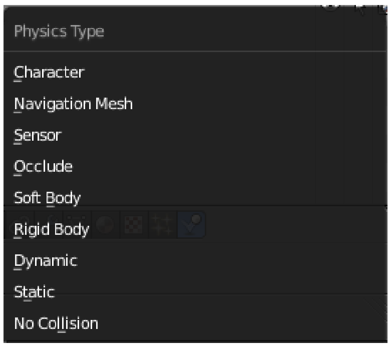
The Blender Game Engine (BGE) is based on Rigid-Body Physics, which differs significantly from the complementary set of tools available in the form of Soft Body Physics Simulations. Though the BGE does have a Soft Body type, it is not nearly as nuanced as the non-BGE Soft Body. The inverse is even more true: it is difficult to get the non-BGE physics to resemble anything like a stiff shape. Rigid Body Physics does not have, as an effect or a cause, any mesh deformations. For a discussion on how to partially overcome this, see: Mesh Deformations.

Global Options

The global Physics Engine settings can be found in the World Properties, which include the Gravity constant and some important engine performance tweaks.

Object Physics

Physics Type



No Collision
Static
Dynamic
Rigid Body
Soft Body
Character Controller
Vehicle Controller
Occluder - Prevents calculation of rendered objects (not their physics, though!).
Sensor- Detects presence without restituting collisions.
Navigation Mesh - To make pathfinding paths. Useful for Artificial Intelligence.

Material Physics

Physics can be associated with a material on the material properties tab. These are settings that one would normally associate with a material, such has it's friction and they are meant to be used in conjunction with the object physics settings, not replace it.

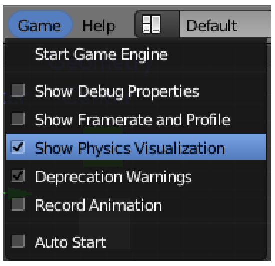
Constraints

It is imperative to understand that the Blender Constraints generally don't work inside the BGE. This means interesting effects such as Copy Rotation are unavailable directly.

Your options include:

- Parenting - But not Vertex Parenting.
- Rigid Body Joint - This is the one Constraint that you can set up through the UI that works in the BGE. It has several options, and can be very powerful - see ITS page for a detailed description and demo .blend. Don't forget that you can loop through objects using `bpy` instead of clicking thousands of times to set up chains of these Constraints.
- Rigid Body Joints on the Fly - You can add/remove them after the BGE starts by using `bge.constraints.createConstraint()`. This can be good either to simply automate their setup, or to truly make them dynamic. A simple demo can be viewed in: BGE-Physics-DynamicallyCreateConstraint.blend
- Python Controllers - As always, in the BGE, you can get the most power when you drop into Python and start toying with the settings directly. For instance, the Copy Rotation mentioned above is not hard -- All you have to do is something to the effect of `own.worldOrientation = bge.logic.getCurrentScene().objects['TheTargetObject'].worldOrientation`

Visualizing Physics



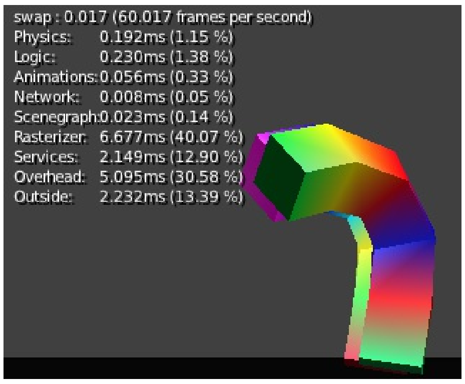
Go to Game » Show Physics Visualization to show lines representing various attributes of the Bullet representation of your objects. Note that these might be easier to see when you turn on Wireframe Mode (Z) before you press P. Also note that you can see how the Bullet triangulation is working (it busts all your Quads to Tris at run-time, but the BGE meshes are still quads at run-time).

- **RGB/XYZ Widget** - Representing the object's Local Orientation and Origin.
- **Green** - "sleeping meshes" that are not moving, saving calculations until an external event "wakes" it.
- **White** - White lines represent active bounding meshes at are undergoing physics calulations, untill such calculations are so small that the object is put to rest. This is how you can see the effects of the Collision Bounds.
    - **Thick, or Many White Lines** - A compound collision mesh/meshes.
- **Violet** - Bounding meshes for Soft bodies.
- **Red** - The Bounding Box, the outer boundary of object. It is always aligned with global X Y and Z, and is used to optimize calculations. Also represents meshes that have been forced into "no sleep" status.
- **Yellow** - Normals.
- **Black** - When in wireframe, this is your mesh's visual appearance.

If you want finer-grained control over the display options, you can add this as a Python Controller and uncomment whichever pieces you want to see:

```
import bge
debugs = (
#bge.constraints.DBG_DRAWWIREFRAME, # Draw wireframe in debug.
bge.constraints.DBG_DRAWAABB, # Draw Axis Aligned Bounding Box in debug.
#bge.constraints.DBG_DRAWFREATURESTEXT, # Draw features text in debug.
#bge.constraints.DBG_DRAWCONTACTPOINTS, # Draw contact points in debug.
#bge.constraints.DBG_NOHELPTEXT, # Debug without help text.
#bge.constraints.DBG_DRAWTEXT, # Draw text in debug.
#bge.constraints.DBG_PROFILETIMINGS, # Draw profile timings in debug.
#bge.constraints.DBG_ENABLESATCOMPARISION, # Enable sat comparision in debug.
#bge.constraints.DBG_DISABLEBULLETLCP, # Disable Bullet LCP.
#bge.constraints.DBG_ENABLECCD, # Enable Continous Colision Detection in debug.
#bge.constraints.DBG_DRAWCONSTRAINTS, # Draw constraints in debug.
#bge.constraints.DBG_DRAWCONSTRAINTLIMITS, # Draw constraint limits in debug.
#bge.constraints.DBG_FASTWIREFRAME, # Draw a fast wireframe in debug.
#bge.constraints.POINTTOPOINT_CONSTRAINT,
#bge.constraints.LINEHINGE_CONSTRAINT,
#bge.constraints.ANGULAR_CONSTRAINT,
#bge.constraints.CONETWIST_CONSTRAINT,
#bge.constraints.VEHICLE_CONSTRAINT,
)
for d in debugs:
  bge.constraints.setDebugMode(d)
```

Show Framerate and Profile



```
swap : 0.017 (60.017 frames per second)
Physics:    0.192ms (1.15 %)
Logic:      0.230ms (1.38 %)
Animations: 0.056ms (0.33 %)
Network:    0.008ms (0.05 %)
Scenegraph: 0.023ms (0.14 %)
Rasterizer: 6.677ms (40.07 %)
Services:   2.149ms (12.90 %)
Overhead:   5.095ms (30.58 %)
Outside:    2.232ms (13.39 %)
```

A shot of Manual-BGE-Physics-DancingSticks.blend with Game » Show Framerate and Profile enabled

If you enable Game » Show Framerate and Profile, it will put some statistics in the upper-left area of the game window.

These can be very informative, but also a bit cryptic. Moguri has elaborated on their meanings, for us: http://mogurijin.wordpress.com/2012/01/03/bge-profile-stats-and-what-they-mean/

Mesh Deformations

As mentioned above, Rigid Body physics do not affect mesh deformations, nor do they account for them in the physics model. This leaves you with a few options:

Soft Bodies

You can try using a Soft Body, but these are fairly hard to configure well.

Actions

To use an Action Actuator to do the deformation, you have to make a choice. If you use Shapekeys in the Action, you will be fine as far as the overall collisions (but see below for the note on `reinstancePhysicsMesh()`). The mesh itself is both a display and a physics mesh, so there is not much to configure.

To use an Armature as the deformer will require a bit of extra thought and effort. Basically the Armature will only deform a mesh if the Armature is the parent of that mesh. But at that point, your mesh will lose its physics responsivenes, and only hang in the air (it's copying the location/rotation of the Armature). To somewhat fix this you can then parent the Armature to a collision mesh (perhaps a simple box or otherwise very-low-poly mesh). This "Deformation Mesh" will be the physics representative, being type: Dynamic or Rigid Body, but it will be set to Invisible. Then "Display Mesh" will be the opposite set to type: No Collision, but visible. This still leaves us with the problem mentioned in the previous paragraph.

When you deform a display mesh, it does not update the corresponding physics mesh. You can view this evidently when you Enable Physics Visualization - the collision bounds will remain exactly as when they began. To fix this, you must call `own.reinstancePhysicsMesh()` in some form. Currently this only works on Triangle Mesh bounds, not Convex Hull. We have prepared a demonstration file in Manual-BGE-Physics-DancingSticks.blend. Note that we had to increase the World » Physics » Physics Steps » Substeps to make the collisions work well. The more basic case is the case the Shapekeyed Action, which you can see in the back area of the scene. Since it is the only object involved, you can call `reinstancePhysicsMesh()` unadorned, and it will do the right thing.

The more complicated case is the Collision Mesh » Armature » Display Mesh cluster, which you can see in the front of the scene. What it does in the .blend is call `reinstancePhysicsMesh(viz)`, that is, passing in a reference to the visual mesh. If we tried to establish this relationship without the use of Python, we would find that Blender's dependency check system would reject it as a cyclic setup. This is an example of where Blender's checking is too coarsely-grained, as this circle is perfectly valid: the grandparent object (the Collision Mesh) controls the location/rotation, while the middle object (the Armature) receives the animated Action, where the child (the Display Mesh) receives the deformation, and passes that on up to the top, harmlessly. Something to note is that the Collision Mesh is merely a plane -- that is all it requires for this, since it will be getting the mesh data from `viz`.

Ragdolls

A third option is to create your items out of many sub-objects, connected together with Rigid Body Joints or similar. This can be quite a bit more work, but the results can be much more like a realistic response to collisions. For an Addon that can help you out in the process, check out the Blender Ragdoll Implementation Kit.

Digging Deeper

Sometimes you will want to look at:

- The main Bullet Physics page - http://bulletphysics.org/wordpress/
- The Bullet Wiki - http://www.bulletphysics.org/mediawiki-1.5.8/index.php?title=Documentation
- The Bullet API Docs - http://www.continuousphysics.com/Bullet/BulletFull/index.html
- The Bullet Forums - http://www.bulletphysics.org/Bullet/phpBB3/

Then there is always:

Reading the Blender and Bullet Source Files

This might sound intimidating, even if you know C/C++, but it can be very informative. You can see how Blender sets up the objects to pass to Bullet, add `printf()`s in places, or otherwise experiment and `svn revert` to get back to normalcy.

Here is an example of the trail to get to the bottom of the handling of the options. We will observe the handling of the `use_shape_match` property, as an example.

- Start by getting The Blender Source Tree
- If you search it for `use_shape_match` (e.g., by `grep -r use_shape_match .`), this will lead you to blender/source/blender/makesrna/intern/rna_object_force.c, which says:

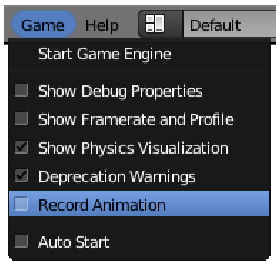
```
prop = RNA_def_property(srna, "use_shape_match", PROP_BOOLEAN, PROP_NONE);
RNA_def_property_boolean_sdna(prop, NULL, "flag", OB_BSB_SHAPE_MATCHING);
RNA_def_property_ui_text(prop, "Shape Match", "Enable soft body shape matching goal");
```

- From this we see that the internal flag is set from the value of `OB_BSB_SHAPE_MATCHING`
- Searching for that leads us to:
  - Its simple initialization in blender/blenkernel/intern/bullet.c
  - Its assignment to `objprop.m_gamesoftFlag`, an object of type `KX_ObjectProperties`, in gameengine/Converter/BL_BlenderDataConversion.cpp -- so far, only passing the value, no actual decision-making.
- Searching for that leads us to gameengine/Physics/Bullet/CcdPhysicsController.cpp where we can find the following:

```
if (m_cci.m_gamesoftFlag & CCD_BSB_SHAPE_MATCHING)//OB_SB_GOAL)
{
    psb->setPose(false,true);//
} else
{
    psb->setPose(true,false);
}
```

- Here is the first bit of logic. It inverts the arguments to `setPose` depending on the value. Now then, since `psb` is of type `btSoftBody`, we have officially launched into Bullet territory. You have a couple options:
    - If you go to the Bullet API Navigator and expand the Class List menu, you can CtrlF for the `btSoftBody` class, and follow the link to the btSoftBody Class Reference Page. There you will see very sparse written documentation, but it will, at least, link you to a syntax-highlighted line where the method is implemented.
    - Get the Bullet Source with: `svn checkout http://bullet.googlecode.com/svn/trunk/ bullet-read-only` and probably run something like `ctags -r .` from that tree every now and then to build the `tags` file. Now you can dig further. Something like `vim -t setPose` will lead you to the implementation in src/BulletSoftBody/btSoftBody.cpp (which is the same code as can be found through the Bullet API Navigator in the previous step).
- Through either approach, we find that the mysterious `bool`s above are for `btSoftBody::setPose(bool bvolume,bool bframe)`, which are immediately assigned to `m_pose.m_bvolume` and `m_pose.m_bframe`, respectively.
    - Subsequently searching for `m_bvolume` doesn't show much use in this file, other than the assignment and initialization. We could follow the trail deeper to the btSoftBody::Pose Struct Reference docs, but for now let's try:
    - Searching for `m_pose.m_bframe`. At this point, in this file, we have finally found the end of the simple passing of the flags, and we will see major chunks of code that are branched depending on this setting.
- Whether we can learn anything apparent at this point will depend on our ability to understand the code and concepts within the Bullet implementation. Perhaps we followed a multi-step process to find inscrutability, but at least we can see the very lines executed within the BGE.
    - Now we have some symbols to search for in Google or in the Bullet Forums.
    - If we wanted to instrument this code with debugging `printf()`s, we could compile it and link it into our Blender build.

Recording to Keyframes



Menu to record Keyframes to the Dopesheet.

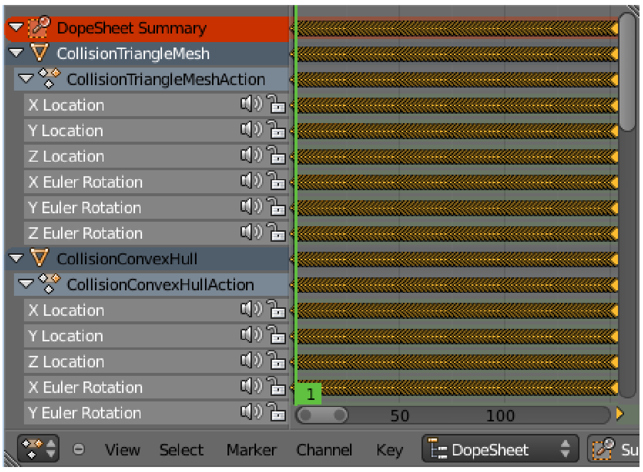
Beyond gaming, sometimes you wish to render a complex scene that involves collisions, multiple forces, friction between multiple bodies and air drag or even a simple setup that is just easier to achieve using the realtime physics.

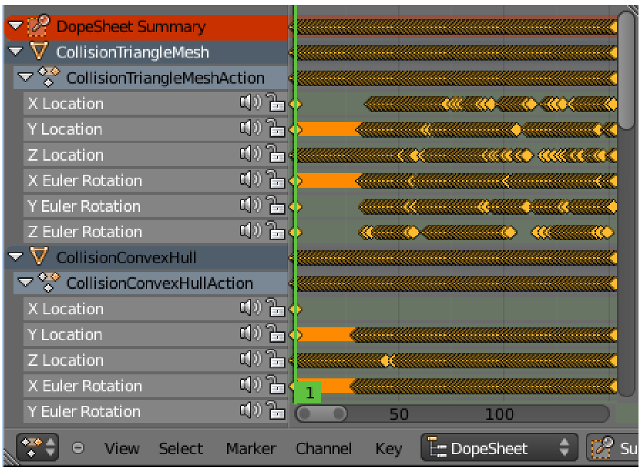
Blender provides a way to *bake* or *record* a physics simulation into keyframes allowing it then to be played as an action either for animation or games. Keep in mind that the result of this method is a recording, no longer a simulation. This means that the result is completely deterministic (the same everytime it is run) and unable to interact with new objects that are added to the physics simulation after it was recorded. This may, or not, be desired according to the situation.

All you have to do to achieve this effect is go to the Info Editor (the bar at the top of the window) Game » Record Animation, and it will lock away your keyframes for use in Blender Render mode. You can go back to the 3D view and hit AltA to play it back, or CtrlF12 to render it out as an animation.

Note that, you can also use Game Logic Bricks and scripting. Everything will be recorded.

Keyframe Clean-up





Record Animation keys redundant data (data that was did not change relative to the last frame). Pressing O while in the DopeSheet will remove all superfluous keyframes. Unwanted channels can also be removed.

Exporting

.bullet / Bullet compatible engines

You can snapshot the physics world at any time with the following code:

```
import bge
bge.constraints.exportBulletFile("test.bullet")
```

This will allow importing into other Bullet-based projects. See the Bullet Wiki on Serialization for more.

Page status (reviewing guidelines)

Void page
Proposed fixes: none

No Collision Physics Object Type

"No Collision" objects in the Game Engine are completely unaffected by Physics, and do cause physics reactions. They are useful as pure display objects, such as the child of a Custom Collision Hull.

Options

The only option available on No Collision types is: Doc:2.6/Manual/Game Engine/Physics/InvisibleOption

All Types

Summary - No Collision | Static | Dynamic | Rigid Body | Soft Body | Occlude | Sensor | Navigation Mesh | Character | Vehicle

Static Physics Object Type

Static objects in the Blender Game Engine do not automatically react to physics, including gravity and collisions. Even if hit by the force of a speeding 18-wheeler truck, it will remain unresponsive in terms of location, rotation, or deformation.

It will, however, give collision reactions. Objects will bounce off of Static Objects, and rotational inertia will transfer to objects capable of rotating (that is, Rigid Body Objects will spin in response, though Dynamic Objects will not).

Note that none of this prevents you from transforming the Static Objects with Logic Bricks or Python code. The visual objects will correctly move and their physics representation will update in the engine as well.

Another important note is that the default Collision Bounds is a Triangle Mesh, meaning it is higher in computational requirements but also in detail. This in turn means the "Radius" option has no effect by default.

In the example game demo, Frijoles, the Static type is represented by the "Arena" object that holds all the moving bits.

For more documentation, see the Top BGE Physics page.

Options

bpy Access

Note that most of these properties are accessible through the non-BGE scripting API via `bpy.data.objects["ObjectName"].game`, which is of type `bpy.types.GameObjectSetting`. This is useful so you can, for example, set a range of objects to have gradated values via a for-loop.

- Actor - Enables detection by Near and Radar Sensors.
  - Default: On.
  - Python property: `obj.game.use_actor`
- Ghost - Disables collisions completely, similar to No Collision.
  - Default: Off.
  - Python property: `obj.game.use_ghost`
- Invisible - Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner).
  - Demo: The "ClothCatcher" object in top of Frijoles.blend
  - Default: Off.
  - Python property: `obj.use_render`

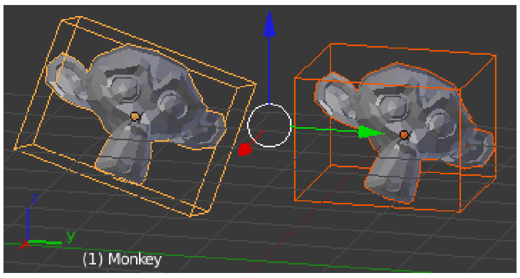
- Radius - If you have the "Collision Bounds: Sphere" set explicitly (or implicitly through having the Collision Bounds subpanel unchecked), this will multiply with the Object's (unapplied) Scale. Note that none of the other bounds types are affected. Also note that in the 3D View the display will show this for all types, even though it is only actually used with Sphere.
  - Range: 0.01-10.
  - Default: 1.
  - Python property: `obj.game.radius`
  - Demo: The table below describes the results visible in Manual-BGE-Physics_BoundsRadiusAndScale.blend.

Basic	Radius=1.5	Unapplied Scale	Applied Scale	Collision Bounds
Rolls, radius of 1 BU	Rolls, radius of 1.5 BU (after "popping" upward)	Rolls, radius of 1.5 BU	Rolls, radius of 1 BU (!)	Default (which is Sphere)
Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1.5 BU	Slides, extent of 1.5 BU	Box
""	""	""	""	Convex Hull
Slides, extent of 1 BU (but with more friction than above)	Slides, extent of 1 BU (but with more friction than above)	Acts insane	Slides, extent of 1.5 BU	Triangle Mesh

- Anisotropic Friction - Isotropic friction is identical at all angles. Anisotropic is directionally-dependant. Here you can vary the coefficients for the three axes individually, or disable friction entirely.
  - Range: 0.1-1.
  - Default: 1.
  - Python properties: `obj.game.use_anisotropic_friction` (boolean) and `obj.game.friction_coefficients` (a 3-element array).

Collision Bounds

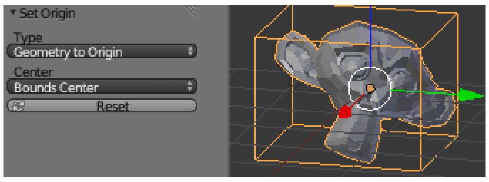
Note: The Static type differs from the others in that it defaults to a Triangle Mesh bounds, instead of a simple sphere.



Demonstration of a Local Bounding Box (left) and a Global Bounding Box (right). The monkeys are identical, except the right one has had its rotation applied with CtrlAR.

The first thing you must understand is the idea of the 3d Bounding Box. If you run through all the vertices of a mesh and record the lowest and highest x values, you have found the x min/max---the complete boundary for all x values within the mesh. Do this again for y and z, then make a rectangular prism out of these values, and you have a Bounding Box. This box could be oriented relative globally to the world or locally to the object's rotation.

The x extent, then, is half of the distance between the x min/max.



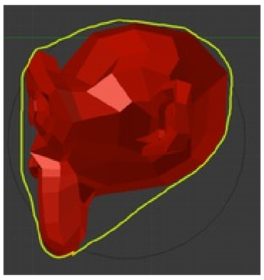
Setting the origin to Bounds Center instead of Median Center.

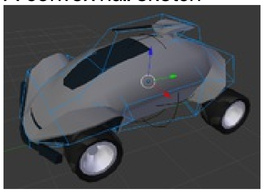
Throughout all of this you must be cognizant of the Object Origin. For the Game engine, the default CtrlAlt⇧ ShiftC,3 (Set Origin » Origin to Geometry) is unlikely to get the desired placement of the Collision Bounds that you want. Instead, you should generally set the origin by looking at the T-toolshelf after you do the Set Origin, and changing the Center from Median Center to Bounds Center. Blender will remember this change for future CtrlAlt⇧ ShiftC executions.

All Collision Bounds are centered on this origin. All boxes are oriented locally, so object rotation matters.

A final introductory comment: When you set the Collision Bounds on an object, Blender will attempt to display a visualization of the bounds in the form of a dotted outline. Currently, there is a bug: The 3D View does not display this bounds preview where it actually will be during the game. To see it, go to Game » Show Physics Visualization and look for the white (or green, if sleeping) geometry.

Now we can explain the various options for the Collision Bounds settings:


A convex hull sketch


Another way to create Collision Bounds -- By hand.

- (Default)
  - For Dynamic and Static objects, it is a Triangle Mesh (see below).
  - For everything else, it is a Sphere (see below).
- Capsule - A cylinder with hemispherical caps, like a pill.
  - Radius of the hemispheres is the greater of the x or y extent.
  - Height is the z bounds
- Box - The x,y,z bounding box, as defined above.
- Sphere -
  - Radius is defined by the object's scale (visible in the N properties panel) times the physics radius (can be found in Physics » Attributes » Radius.
  - Note: This is the only bounds that respects the Radius option.

- Cylinder
  - Radius is the greater of the x or y extent.
  - Height is the z bounds.
- Cone
  - Base radius is the greater of the x or y extent.
  - Height is the z bounds.
- Convex Hull - Forms a shrink-wrapped, simplified geometry around the object.
  - For the math, see Wikipedia's entry on Convex Hull or Wolfram's entry on Convex Hull.
  - For a demo, see the image to the right, where we have sketched a hull around Suzanne's profile:
- Triangle mesh - Most expensive, but most precise. Collision will happen with all of triangulated polygons, instead of using a virtual mesh to approximate that collision.
- By Hand - This is not an option in the Physics tab's Collision Bounds settings, but a different approach, entirely. You create a second mesh, which is invisible, to be the physics representation. This becomes the parent for your display object. Then, your display object is set to ghost so it doesn't fight with the parent object. This method allows you to strike a balance between the accuracy of Triangle Mesh with the efficiency of some of the others. See the demo of this in the dune buggy to the right.

Options

There are only two options in the Collision Bounds subpanel.

- Margin - "Add extra margin around object for collision detection, small amount required for stability." If you find your objects are getting stuck in places they shouldn't, try increasing this to, say, 0.06.
  - Sometimes 0.06 is the default (such as on the Default Cube), but sometimes it is not. You have to keep an eye on the setting, or else learn the symptoms so you can respond when it gives you trouble.
  - You can see somewhat of a demo in Manual-BGE-Physics_Margins.blend. Here, the 0.0 settings are not clearly wrong, but they are different from the 0.06 settings. The 1.0 are simply wrong for the Box and Convex Hull objects. As you will notice, as of 2.62, the Margin has an odd effect on "Sphere" Collision Bounds types. It is almost imperceptibly different - so you will have to see the System Console to view the z-axis measurement. When you look at it, you will find that the purple row of Sphere bounds objects behave nearly identically in spite of the varying Margins.
  - The range is 0.0-1.0
  - If you're lazy/paranoid/unsure/diligent/bored, you can always run this on the Python Console to bump all 0.0 margins to 0.06: `for obj in bpy.data.objects: obj.game.collision_margin = obj.game.collision_margin or 0.06`
- Compound - "Add children to form compound collision object." Basically, if you have a child object and do not have this enabled, the child's collisions will not have an effect on that object "family" (though it will still push other objects around). If you do have it checked, the parent's physics will respond to the child's collision (thus updating the whole family). For a demonstration, look at the far right of the top row of Manual-BGE-Physics-CollisionBounds.blend. In it, the Empty.001 has "Compound" unchecked, and so it falls down, while Empty.001 has it checked, and behaves properly. Python property: `obj.game.use_collision_compound`

Example Demo

To see working examples of most of the types of Collision Bounds configurations, see Manual-BGE-Physics-CollisionBounds.blend. The objects in red have some kind of flawed setting, and the green ones are the improved versions. It already has the Show Physics Visualization setting checked, so when you hit P you will see the bounds behavior in white wireframes.

Here you will see:

- First Row:
  - A rotated Cube, with and without "Collision Bounds" checked (demonstrates that the default bounds does not rotate with the object rotation).
  - The difference between setting object origins to the default "Median" center versus the "Bounds" center.
  - Suzanne falling onto a custom shelf, perfectly made for her jaw shape. Only Triangle Mesh results in good behavior for this one.
  - Options:
    - (Excessively) increased Margin.
    - Compound when parenting is involved.
- Second Row:
  - All 7 incorrect types of bounds (for this case).
  - 3 correct types (Convex Hull, Triangle Mesh, and custom).

Origin, Rotation, Scale

You must understand Applying Rotation/Scale with CtrlA and setting object Origins with CtrlAlt⇧ ShiftC. As mentioned in the Collision Bounds Section, the default behavior for Set Origin is to put it at the Median Center, but for the BGE you will usually want to use Bounds Center.

Some examples, as demonstrated in: Manual-BGE-Physics-TransformApply.blend

- An object's bounds are defined by the min/max of all the vertices for each of the axes. The center of gravity is defined by its Origin (the orange dot in the 3d View).
- Rotating does not affect the default collision bounds -- Collision Bounds option must enabled if you want to explicitly set the type of Collision Bounds so that you can choose a type where rotation has an affect.
- Scaling *does* affect the object's bounds effective radius value, but if you CtrlAS (Apply Scale), the bounds pop back out to a larger size. You must set the Collision Bounds (or the Radius) explicitly if you want a more correct physics representation.

Create Obstacle

to do

- It has to do with pathfinding and obstacle avoidance.
- from the recent Recast (generating navmeshes) and Detour (navmesh/pathfinding) additions: http://code.google.com/p/recastnavigation/
- World > Obstacle simulation

All Types

Summary - No Collision | Static | Dynamic | Rigid Body | Soft Body | Occlude | Sensor | Navigation Mesh | Character | Vehicle

Dynamic Physics Object Type

Dynamic objects in the Game Engine give/receive collisions, but when they do so they themselves do not rotate in response. So, a Dynamic ball will hit a ramp and slide down, while a Rigid Body ball would begin rotating.

If you do not need the rotational response the Dynamic type can save the extra computation.

Note that these objects can still be rotated with Logic Bricks or Python code. Their physics meshes will update when you do these rotations - so collisions will be based on the new orientations.

In the example game demo, Frijoles, the Dynamic type is represented by the titular jumping beans. Though we want these characters to recoil back when they hit a Boulder or each other, having them torque in response to these collisions would result in their being impossible to control.

For more documentation, see the Top BGE Physics page.

Options

bpy Access

Note that most of these properties are accessible through the non-BGE scripting API via `bpy.data.objects["ObjectName"].game`, which is of type `bpy.types.GameObjectSetting`. This is useful so you can, for example, set a range of objects to have gradated values via a for-loop.

- Actor - Enables detection by Near and Radar Sensors.
  - Default: On.
  - Python property: `obj.game.use_actor`
- Ghost - Disables collisions completely, similar to No Collision.
  - Default: Off.
  - Python property: `obj.game.use_ghost`
- Invisible - Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner).
  - Demo: The "ClothCatcher" object in top of Frijoles.blend
  - Default: Off.
  - Python property: `obj.use_render`Doc:2.6/Manual/Game Engine/Physics/RightColumnOfOptions

- Mass - Affects the reaction due to collision between objects -- more massive objects have more inertia. Will also affect material force fields. Will also change behaviors if you are using the suspension and steering portions of Bullet physics.
  - Demo: The three different Boulder sizes in Frijoles.blend
  - Range: 0.01-10,000.
  - Default: 1.
  - Python property: `obj.game.mass`
- Radius - If you have the "Collision Bounds: Sphere" set explicitly (or implicitly through having the Collision Bounds subpanel unchecked), this will multiply with the Object's (unapplied) Scale. Note that none of the other bounds types are affected. Also note that in the 3D View the display will show this for all types, even though it is only actually used with Sphere.
  - Range: 0.01-10.
  - Default: 1.
  - Python property: `obj.game.radius`
  - Demo: The table below describes the results visible in Manual-BGE-Physics_BoundsRadiusAndScale.blend.

Basic	Radius=1.5	Unapplied Scale	Applied Scale	Collision Bounds
Rolls, radius of 1 BU	Rolls, radius of 1.5 BU (after "popping" upward)	Rolls, radius of 1.5 BU	Rolls, radius of 1 BU (!)	Default (which is Sphere)
Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1.5 BU	Slides, extent of 1.5 BU	Box
""	""	""	""	Convex Hull
Slides, extent of 1 BU (but with more friction than above)	Slides, extent of 1 BU (but with more friction than above)	Acts insane	Slides, extent of 1.5 BU	Triangle Mesh

- Form Factor - For affecting the Inertia Tensor. The higher the value, the greater the rotational inertia, and thus the more resistant to torque. You might think this is strange, considering Dynamic types do not have torque in response to collisions -- but you can still see this value's effects when you manually apply Torque.
  - Demo: Manual-BGE-Physics-DynamicFormFactor.blend. The cube on the left has a Form Factor of 0.001, while the one on the right has a Form Factor of 1.0.
  - Range: 0-1.
  - Default: 0.4.
  - Python property: `obj.game.form_factor`

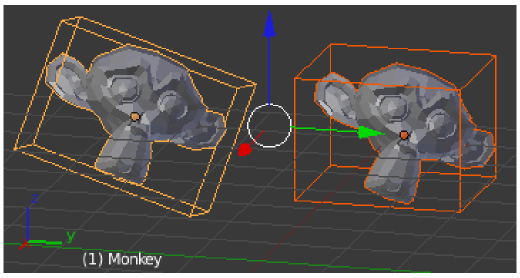
- Anisotropic Friction - Isotropic friction is identical at all angles. Anisotropic is directionally-dependant. Here you can vary the coefficients for the three axes individually, or disable friction entirely.
  - Range: 0.1-1.
  - Default: 1.
  - Python properties: `obj.game.use anisotropic friction` (boolean) and `obj.game.friction coefficients` (a 3-element

array).

- Velocity - Limit the speed of an object. "0" is no limit.
    - Range: 0-1000.
    - Suboption: Minimum - The object is allowed to be at complete rest, but as soon as it accelerates it will immediately jump to the minimum speed.
        - Default: 0.
        - Python property: `obj.game.velocity_min`
    - Suboption: Maximum - Top speed of the object.
        - Default: 0. (Unlimited.)
        - Python property: `obj.game.velocity_max`
- Damping - Increase the "sluggishness" of the object.
    - Range: 0-1.
    - Suboption: Translation - Resist movement. At "1" the object is completely immobile.
        - Range: 0-1.
        - Default: 0.0254.
        - Python property: `obj.game.damping`
    - Suboption: Rotation - Resist rotation, but not the kind of rotation that comes from a collision. For example, if a Motion Controller applies Torque to an object, this damping will be a factor.
        - Range: 0-1.
        - Default: 0.159.
        - Python property: `obj.game.rotation_damping`

- Lock Translation - Seize the object in the world along one or more axes. Note that this is global coordinates, not local or otherwise.
    - Defaults: All off.
    - X - Python property: `obj.game.lock_location_x`
    - Y - Python property: `obj.game.lock_location_y`
    - Z - Python property: `obj.game.lock_location_z`
- Lock Rotation - Same, but for rotation (also with respect to the global coordinates).
    - Defaults: All off.
    - X - Python property: `obj.game.lock_rotation_x`
    - Y - Python property: `obj.game.lock_rotation_y`
    - Z - Python property: `obj.game.lock_rotation_z`

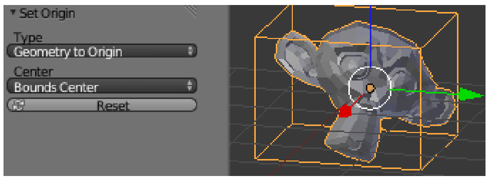
Collision Bounds



Demonstration of a Local Bounding Box (left) and a Global Bounding Box (right). The monkeys are identical, except the right one has had its rotation applied with CtrlAR.

The first thing you must understand is the idea of the 3d Bounding Box. If you run through all the vertices of a mesh and record the lowest and highest x values, you have found the x min/max---the complete boundary for all x values within the mesh. Do this again for y and z, then make a rectangular prism out of these values, and you have a Bounding Box. This box could be oriented relative globally to the world or locally to the object's rotation.

The x extent, then, is half of the distance between the x min/max.



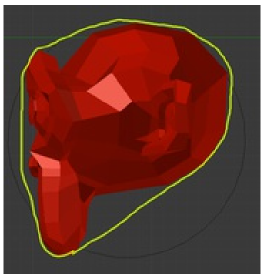
Setting the origin to Bounds Center instead of Median Center.

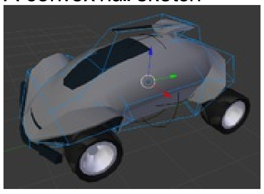
Throughout all of this you must be cognizant of the Object Origin. For the Game engine, the default CtrlAlt⇧ ShiftC,3 (Set Origin » Origin to Geometry) is unlikely to get the desired placement of the Collision Bounds that you want. Instead, you should generally set the origin by looking at the T-toolshelf after you do the Set Origin, and changing the Center from Median Center to Bounds Center. Blender will remember this change for future CtrlAlt⇧ ShiftC executions.

All Collision Bounds are centered on this origin. All boxes are oriented locally, so object rotation matters.

A final introductory comment: When you set the Collision Bounds on an object, Blender will attempt to display a visualization of the bounds in the form of a dotted outline. Currently, there is a bug: The 3D View does not display this bounds preview where it actually will be during the game. To see it, go to Game » Show Physics Visualization and look for the white (or green, if sleeping) geometry.

Now we can explain the various options for the Collision Bounds settings:


A convex hull sketch


Another way to create Collision Bounds -- By hand.

- (Default)
  - For Dynamic and Static objects, it is a Triangle Mesh (see below).
  - For everything else, it is a Sphere (see below).
- Capsule - A cylinder with hemispherical caps, like a pill.
  - Radius of the hemispheres is the greater of the x or y extent.
  - Height is the z bounds
- Box - The x,y,z bounding box, as defined above.
- Sphere -
  - Radius is defined by the object's scale (visible in the N properties panel) times the physics radius (can be found in Physics » Attributes » Radius.
  - Note: This is the only bounds that respects the Radius option.
- Cylinder
  - Radius is the greater of the x or y extent.
  - Height is the z bounds.
- Cone
  - Base radius is the greater of the x or y extent.
  - Height is the z bounds.
- Convex Hull - Forms a shrink-wrapped, simplified geometry around the object.
  - For the math, see Wikipedia's entry on Convex Hull or Wolfram's entry on Convex Hull.
  - For a demo, see the image to the right, where we have sketched a hull around Suzanne's profile:
- Triangle mesh - Most expensive, but most precise. Collision will happen with all of triangulated polygons, instead of using a virtual mesh to approximate that collision.
- By Hand - This is not an option in the Physics tab's Collision Bounds settings, but a different approach, entirely. You create a second mesh, which is invisible, to be the physics representation. This becomes the parent for your display object. Then, your display object is set to ghost so it doesn't fight with the parent object. This method allows you to strike a balance between the accuracy of Triangle Mesh with the efficiency of some of the others. See the demo of this in the dune buggy to the right.

Options

There are only two options in the Collision Bounds subpanel.

- Margin - "Add extra margin around object for collision detection, small amount required for stability." If you find your objects are getting stuck in places they shouldn't, try increasing this to, say, 0.06.
  - Sometimes 0.06 is the default (such as on the Default Cube), but sometimes it is not. You have to keep an eye on the setting, or else learn the symptoms so you can respond when it gives you trouble.
  - You can see somewhat of a demo in Manual-BGE-Physics_Margins.blend. Here, the 0.0 settings are not clearly wrong, but they are different from the 0.06 settings. The 1.0 are simply wrong for the Box and Convex Hull objects. As you will notice, as of 2.62, the Margin has an odd effect on "Sphere" Collision Bounds types. It is almost imperceptibly different - so you will have to see the System Console to view the z-axis measurement. When you look at it, you will find that the purple row of Sphere bounds objects behave nearly identically in spite of the varying Margins.
  - The range is 0.0-1.0
  - If you're lazy/paranoid/unsure/diligent/bored, you can always run this on the Python Console to bump all 0.0 margins to 0.06: `for obj in bpy.data.objects: obj.game.collision_margin = obj.game.collision_margin or 0.06`
- Compound - "Add children to form compound collision object." Basically, if you have a child object and do not have this enabled, the child's collisions will not have an effect on that object "family" (though it will still push other objects around). If you do have it checked, the parent's physics will respond to the child's collision (thus updating the whole family). For a demonstration, look at the far right of the top row of Manual-BGE-Physics-CollisionBounds.blend. In it, the Empty.001 has "Compound" unchecked, and so it falls down, while Empty.001 has it checked, and behaves properly. Python property: `obj.game.use_collision_compound`

Example Demo

To see working examples of most of the types of Collision Bounds configurations, see Manual-BGE-Physics-CollisionBounds.blend. The objects in red have some kind of flawed setting, and the green ones are the improved versions. It already has the Show Physics Visualization setting checked, so when you hit P you will see the bounds behavior in white wireframes.

Here you will see:

- First Row:
  - A rotated Cube, with and without "Collision Bounds" checked (demonstrates that the default bounds does not rotate with the object rotation).
  - The difference between setting object origins to the default "Median" center versus the "Bounds" center.
  - Suzanne falling onto a custom shelf, perfectly made for her jaw shape. Only Triangle Mesh results in good behavior for this one.
  - Options:
    - (Excessively) increased Margin.
    - Compound when parenting is involved.
- Second Row:
  - All 7 incorrect types of bounds (for this case).
  - 3 correct types (Convex Hull, Triangle Mesh, and custom).

Origin, Rotation, Scale

You must understand Applying Rotation/Scale with CtrlA and setting object Origins with CtrlAlt⇧ ShiftC. As mentioned in the Collision Bounds Section, the default behavior for Set Origin is to put it at the Median Center, but for the BGE you will usually want to use Bounds Center.

Some examples, as demonstrated in: Manual-BGE-Physics-TransformApply.blend

- An object's bounds are defined by the min/max of all the vertices for each of the axes. The center of gravity is defined by its Origin (the orange dot in the 3d View).
- Rotating does not affect the default collision bounds -- Collision Bounds option must enabled if you want to explicitly set the type of Collision Bounds so that you can choose a type where rotation has an affect.
- Scaling *does* affect the object's bounds effective radius value, but if you CtrlAS (Apply Scale), the bounds pop back out to a larger size. You must set the Collision Bounds (or the Radius) explicitly if you want a more correct physics representation.

Create Obstacle

to do

- It has to do with pathfinding and obstacle avoidance.
- from the recent Recast (generating navmeshes) and Detour (navmesh/pathfinding) additions: http://code.google.com/p/recastnavigation/
- World > Obstacle simulation

All Types

Summary - No Collision | Static | Dynamic | Rigid Body | Soft Body | Occlude | Sensor | Navigation Mesh | Character | Vehicle

Rigid Body Physics Object Type

Probably the most common type of object in the Game Engine. It will give/receive collisions and react with a change in its velocity and its rotation. A Rigid Body ball would begin rotating and roll down (where a Dynamic ball would only hit and slide down the ramp).

The idea behind Rigid Body dynamics is that the mesh does not deform. If you need deformation you will need to either go to Soft Body or else fake it with animated Actions.

In the example game demo, Frijoles, the Rigid Body type is represented by the Boulders that spawn from the top of the level. Notice how they tumble and roll in response to the collisions with the Arena.

For more documentation, see the Top BGE Physics page.

Options

bpy Access

Note that most of these properties are accessible through the non-BGE scripting API via `bpy.data.objects["ObjectName"].game`, which is of type `bpy.types.GameObjectSetting`. This is useful so you can, for example, set a range of objects to have gradated values via a for-loop.

- Actor - Enables detection by Near and Radar Sensors.
  - Default: On.
  - Python property: `obj.game.use_actor`
- Ghost - Disables collisions completely, similar to No Collision.
  - Default: Off.
  - Python property: `obj.game.use_ghost`
- Invisible - Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner).
  - Demo: The "ClothCatcher" object in top of Frijoles.blend
  - Default: Off.
  - Python property: `obj.use_render`

- Mass - Affects the reaction due to collision between objects -- more massive objects have more inertia. Will also affect material force fields. Will also change behaviors if you are using the suspension and steering portions of Bullet physics.
  - Demo: The three different Boulder sizes in Frijoles.blend
  - Range: 0.01-10,000.
  - Default: 1.
  - Python property: `obj.game.mass`
- Radius - If you have the "Collision Bounds: Sphere" set explicitly (or implicitly through having the Collision Bounds subpanel unchecked), this will multiply with the Object's (unapplied) Scale. Note that none of the other bounds types are affected. Also note that in the 3D View the display will show this for all types, even though it is only actually used with Sphere.
  - Range: 0.01-10.
  - Default: 1.
  - Python property: `obj.game.radius`
  - Demo: The table below describes the results visible in Manual-BGE-Physics_BoundsRadiusAndScale.blend.

Basic	Radius=1.5	Unapplied Scale	Applied Scale	Collision Bounds
Rolls, radius of 1 BU	Rolls, radius of 1.5 BU (after "popping" upward)	Rolls, radius of 1.5 BU	Rolls, radius of 1 BU (!)	Default (which is Sphere)
Slides, extent of 1 BU	Slides, extent of 1 BU	Slides, extent of 1.5 BU	Slides, extent of 1.5 BU	Box
""	""	""	""	Convex Hull
Slides, extent of 1 BU (but with more friction than above)	Slides, extent of 1 BU (but with more friction than above)	Acts insane	Slides, extent of 1.5 BU	Triangle Mesh

- Form Factor - For affecting the Inertia Tensor. The higher the value, the greater the rotational inertia, and thus the more resistant to torque. You might think this is strange, considering Dynamic types do not have torque in response to collisions -- but you can still see this value's effects when you manually apply Torque.
  - Demo: Manual-BGE-Physics-DynamicFormFactor.blend. The cube on the left has a Form Factor of 0.001, while the one on the right has a Form Factor of 1.0.
  - Range: 0-1.
  - Default: 0.4.
  - Python property: `obj.game.form_factor`

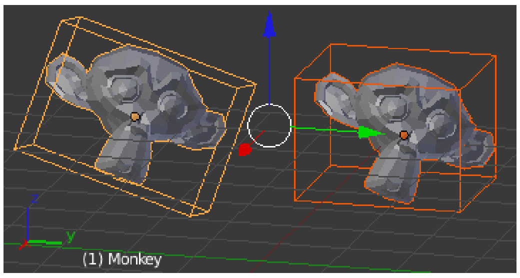
- Anisotropic Friction - Isotropic friction is identical at all angles. Anisotropic is directionally-dependant. Here you can vary the coefficients for the three axes individually, or disable friction entirely.
  - Range: 0.1-1.
  - Default: 1.
  - Python properties: `obj.game.use_anisotropic_friction` (boolean) and `obj.game.friction_coefficients` (a 3-element array).

- Velocity - Limit the speed of an object. "0" is no limit.

  - Range: 0-1000.
  - Suboption: Minimum - The object is allowed to be at complete rest, but as soon as it accelerates it will immediately jump to the minimum speed.
    - Default: 0.
    - Python property: `obj.game.velocity_min`
  - Suboption: Maximum - Top speed of the object.
    - Default: 0. (Unlimited.)
    - Python property: `obj.game.velocity_max`
- Damping - Increase the "sluggishness" of the object.
  - Range: 0-1.
  - Suboption: Translation - Resist movement. At "1" the object is completely immobile.
    - Range: 0-1.
    - Default: 0.0254.
    - Python property: `obj.game.damping`
  - Suboption: Rotation - Resist rotation, but not the kind of rotation that comes from a collision. For example, if a Motion Controller applies Torque to an object, this damping will be a factor.
    - Range: 0-1.
    - Default: 0.159.
    - Python property: `obj.game.rotation_damping`

- Lock Translation - Seize the object in the world along one or more axes. Note that this is global coordinates, not local or otherwise.
  - Defaults: All off.
  - X - Python property: `obj.game.lock_location_x`
  - Y - Python property: `obj.game.lock_location_y`
  - Z - Python property: `obj.game.lock_location_z`
- Lock Rotation - Same, but for rotation (also with respect to the global coordinates).
  - Defaults: All off.
  - X - Python property: `obj.game.lock_rotation_x`
  - Y - Python property: `obj.game.lock_rotation_y`
  - Z - Python property: `obj.game.lock_rotation_z`

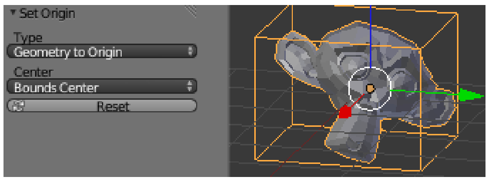
Collision Bounds



Demonstration of a Local Bounding Box (left) and a Global Bounding Box (right). The monkeys are identical, except the right one has had its rotation applied with CtrlAR.

The first thing you must understand is the idea of the 3d Bounding Box. If you run through all the vertices of a mesh and record the lowest and highest x values, you have found the x min/max---the complete boundary for all x values within the mesh. Do this again for y and z, then make a rectangular prism out of these values, and you have a Bounding Box. This box could be oriented relative globally to the world or locally to the object's rotation.

The x extent, then, is half of the distance between the x min/max.



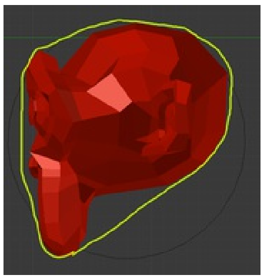
Setting the origin to Bounds Center instead of Median Center.

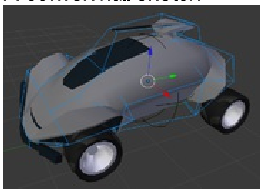
Throughout all of this you must be cognizant of the Object Origin. For the Game engine, the default CtrlAlt⇧ ShiftC,3 (Set Origin » Origin to Geometry) is unlikely to get the desired placement of the Collision Bounds that you want. Instead, you should generally set the origin by looking at the T-toolshelf after you do the Set Origin, and changing the Center from Median Center to Bounds Center. Blender will remember this change for future CtrlAlt⇧ ShiftC executions.

All Collision Bounds are centered on this origin. All boxes are oriented locally, so object rotation matters.

A final introductory comment: When you set the Collision Bounds on an object, Blender will attempt to display a visualization of the bounds in the form of a dotted outline. Currently, there is a bug: The 3D View does not display this bounds preview where it actually will be during the game. To see it, go to Game » Show Physics Visualization and look for the white (or green, if sleeping) geometry.

Now we can explain the various options for the Collision Bounds settings:


A convex hull sketch


Another way to create Collision Bounds -- By hand.

- (Default)
  - For Dynamic and Static objects, it is a Triangle Mesh (see below).
  - For everything else, it is a Sphere (see below).
- Capsule - A cylinder with hemispherical caps, like a pill.
  - Radius of the hemispheres is the greater of the x or y extent.
  - Height is the z bounds
- Box - The x,y,z bounding box, as defined above.
- Sphere -
  - Radius is defined by the object's scale (visible in the N properties panel) times the physics radius (can be found in Physics » Attributes » Radius.
  - Note: This is the only bounds that respects the Radius option.
- Cylinder
  - Radius is the greater of the x or y extent.
  - Height is the z bounds.
- Cone
  - Base radius is the greater of the x or y extent.
  - Height is the z bounds.
- Convex Hull - Forms a shrink-wrapped, simplified geometry around the object.
  - For the math, see Wikipedia's entry on Convex Hull or Wolfram's entry on Convex Hull.
  - For a demo, see the image to the right, where we have sketched a hull around Suzanne's profile:
- Triangle mesh - Most expensive, but most precise. Collision will happen with all of triangulated polygons, instead of using a virtual mesh to approximate that collision.
- By Hand - This is not an option in the Physics tab's Collision Bounds settings, but a different approach, entirely. You create a second mesh, which is invisible, to be the physics representation. This becomes the parent for your display object. Then, your display object is set to ghost so it doesn't fight with the parent object. This method allows you to strike a balance between the accuracy of Triangle Mesh with the efficiency of some of the others. See the demo of this in the dune buggy to the right.

Options

There are only two options in the Collision Bounds subpanel.

- Margin - "Add extra margin around object for collision detection, small amount required for stability." If you find your objects are getting stuck in places they shouldn't, try increasing this to, say, 0.06.
  - Sometimes 0.06 is the default (such as on the Default Cube), but sometimes it is not. You have to keep an eye on the setting, or else learn the symptoms so you can respond when it gives you trouble.
  - You can see somewhat of a demo in Manual-BGE-Physics_Margins.blend. Here, the 0.0 settings are not clearly wrong, but they are different from the 0.06 settings. The 1.0 are simply wrong for the Box and Convex Hull objects. As you will notice, as of 2.62, the Margin has an odd effect on "Sphere" Collision Bounds types. It is almost imperceptibly different - so you will have to see the System Console to view the z-axis measurement. When you look at it, you will find that the purple row of Sphere bounds objects behave nearly identically in spite of the varying Margins.
  - The range is 0.0-1.0
  - If you're lazy/paranoid/unsure/diligent/bored, you can always run this on the Python Console to bump all 0.0 margins to 0.06: `for obj in bpy.data.objects: obj.game.collision_margin = obj.game.collision_margin or 0.06`
- Compound - "Add children to form compound collision object." Basically, if you have a child object and do not have this enabled, the child's collisions will not have an effect on that object "family" (though it will still push other objects around). If you do have it checked, the parent's physics will respond to the child's collision (thus updating the whole family). For a demonstration, look at the far right of the top row of Manual-BGE-Physics-CollisionBounds.blend. In it, the Empty.001 has "Compound" unchecked, and so it falls down, while Empty.001 has it checked, and behaves properly. Python property: `obj.game.use_collision_compound`

Example Demo

To see working examples of most of the types of Collision Bounds configurations, see Manual-BGE-Physics-CollisionBounds.blend. The objects in red have some kind of flawed setting, and the green ones are the improved versions. It already has the Show Physics Visualization setting checked, so when you hit P you will see the bounds behavior in white wireframes.

Here you will see:

- First Row:
    - A rotated Cube, with and without "Collision Bounds" checked (demonstrates that the default bounds does not rotate with the object rotation).
    - The difference between setting object origins to the default "Median" center versus the "Bounds" center.
    - Suzanne falling onto a custom shelf, perfectly made for her jaw shape. Only Triangle Mesh results in good behavior for this one.
    - Options:
        - (Excessively) increased Margin.
        - Compound when parenting is involved.
- Second Row:
    - All 7 incorrect types of bounds (for this case).
    - 3 correct types (Convex Hull, Triangle Mesh, and custom).

Origin, Rotation, Scale

You must understand Applying Rotation/Scale with CtrlA and setting object Origins with CtrlAlt⇧ ShiftC. As mentioned in the Collision Bounds Section, the default behavior for Set Origin is to put it at the Median Center, but for the BGE you will usually want to use Bounds Center.

Some examples, as demonstrated in: Manual-BGE-Physics-TransformApply.blend

- An object's bounds are defined by the min/max of all the vertices for each of the axes. The center of gravity is defined by its Origin (the orange dot in the 3d View).
- Rotating does not affect the default collision bounds -- Collision Bounds option must enabled if you want to explicitly set the type of Collision Bounds so that you can choose a type where rotation has an affect.
- Scaling *does* affect the object's bounds effective radius value, but if you CtrlAS (Apply Scale), the bounds pop back out to a larger size. You must set the Collision Bounds (or the Radius) explicitly if you want a more correct physics representation.

Create Obstacle

to do

- It has to do with pathfinding and obstacle avoidance.
- from the recent Recast (generating navmeshes) and Detour (navmesh/pathfinding) additions: http://code.google.com/p/recastnavigation/
- World > Obstacle simulation

All Types

Summary - No Collision | Static | Dynamic | Rigid Body | Soft Body | Occlude | Sensor | Navigation Mesh | Character | Vehicle

Soft Body Physics Object Type

The most advanced type of object in the Game Engine. Also, it is the most finicky. If you are used to the fun experimentation that comes from playing around with the non-BGE Soft Body sims (such as Cloth), you will probably find a frustrating lack of options and exciting results. Do not despair, we are here to help you get some reasonable settings.

Your setup will involve making sure you have sufficient geometry in the Soft Body's mesh to support the deformation, as well as tweaking the options.

In the example game demo, Frijoles, the Soft Body type is represented by the decorative checkered flag at the top of the level.

For more documentation, see the Top BGE Physics page.

Options

- Actor - Enables detection by Near and Radar Sensors.
  - Default: On.
  - Python property: `obj.game.use_actor`
- Ghost - Disables collisions completely, similar to No Collision.
  - Default: Off.
  - Python property: `obj.game.use_ghost`
- Invisible - Does not display, the same as setting the object to unrendered (such as unchecking the "Camera" icon in the Outliner).
  - Demo: The "ClothCatcher" object in top of Frijoles.blend
  - Default: Off.
  - Python property: `obj.use_render`
- Mass - Affects the reaction due to collision between objects -- more massive objects have more inertia. Will also affect material force fields. Will also change behaviors if you are using the suspension and steering portions of Bullet physics.
  - Demo: The three different Boulder sizes in Frijoles.blend
  - Range: 0.01-10,000.
  - Default: 1.
  - Python property: `obj.game.mass`

- Shape Match - Upon starting the Game Engine this will record the starting shape of the mesh as the "lowest energy" state. This means that the edges will have tension whenever they are flexed to some other form. This is set to on by default, and in this configuration turns the object into more of a thin sheet of metal rather than a cloth.
  - Demo: BGE-Physics-Objects-SoftBodies_ShapeMatchAndLinearStiffness.blend
  - Default: On.
  - Code effect: When on, it will call `btSoftBody::setPose(false,true)`
  - Python property: `obj.game.soft_body.use_shape_match`
  - Suboption: Threshold - Linearly scales the pose match
    - A threshold of 1.0 makes it behave like Shape Match on with a Linear Stiffness of 1.0.
    - A threshold of 0.0 makes it behave like Shape Match off with a Linear Stiffness of 0.0.
    - Range: 0-1.
    - Default: 0.5.
    - Code effect: Sets btSoftBody::Config::kMT.
    - Python property: `obj.game.soft_body.shape_threshold`
- Welding - [Note: Seems to not be hooked up. Blender will tell Bullet to weld any time you enable Soft Body. Look at BL_BlenderDataConversion.cpp where `objprop.m_soft_welding`is hard-coded to 0.0f]
- Position Iteration - Increase the accuracy at a linearly-increasing expense of time. The effect is visible especially with Soft Bodies that fall on sharp corners, though this can slow down even very simple scenes.
  - Demo: A situation where only the max setting of 10 works correctly: BGE-Physics-Objects-SoftBodies_PositionIterations.blend.
  - Range: 0-10.
  - Default: 2.
  - Code effect: Represents the number of times this loop is run.
  - Python property: `obj.game.soft_body.location_iterations`
- Linear Stiffness - Linear stiffness of the soft body links. This is most evident when you have Shape Match off, but it is also evident with it on.
  - Demo: BGE-Physics-Objects-SoftBodies_ShapeMatchAndLinearStiffness.blend
  - Range: 0-1.
  - Default: 0.5.
  - Python property: `obj.game.soft_body.linear_stiffness`
- Friction - Dynamic friction coefficient. TODO: Learn/demo/explain.
  - Code effect: Sets btSoftBody::Config::kMT, which, for Soft Bodies, defines the minimum friction versus the Material Friction (which in turn defaults to 0.5).
  - Range: 0-1.
  - Default: 0.2.
  - Python property: `obj.game.soft_body.dynamic_friction`
- Margin - Small value makes the algorithm unstable. TODO: Learn/demo/explain.
  - Range: 0.01-1.
  - Default: 0.01.
  - Python property: `obj.game.soft_body.collision_margin`
- Bending Constraint - Enable Bending Constraints TODO: Learn/demo/explain.
  - Default: On.
  - Python property: `obj.game.soft_body.use_bending_constraints`
- Cluster Collision - Affects Collision sensors as well as physics.

- Demo: BGE-Physics-Objects-SoftBodies_ClusterRigidToSoftBody.blend for a demonstration of the effect on the Collision Sensor. There you will observe the "Rigid to Soft Body" off, then on with Iterations of 1, 64, and 128. The Off and Iterations: 1 cases do not register collisions, and the other two do (though they send their poor Cubes flying into space).
- Demo of badness: Manual-BGE-Physics-SoftBody_BadClusterCollisions.blend - four different ways of making misconfigured Soft Body objects.
- Suboption: Rigid to Soft Body - Enable cluster collisions between Rigid and Soft Bodies.
  - Default: Off.
  - Python property: `obj.game.soft_body.use_cluster_rigid_to_softbody`
- Suboption: Soft to Soft Body - Enable cluster collisions among Soft Bodies.
  - Default: Off.
  - Python property: `obj.game.soft_body.use_cluster_soft_to_softbody`
- Suboption: Iterations - Number of cluster iterations.
  - Range: 1-128.
  - Default: 64.
  - Python property: `obj.game.soft_body.cluster_iterations`

Hints

- A very important configurable in the case of Soft Body interactions is World properties » Physics » Physics Steps » Substeps. In the test .blend here: Manual-BGE-Physics-SoftBody_PhysicsSteps.blend, you can see the behavior at various Substep levels:
  1. The default level. The Grid object goes straight through the cube, hardly slowing down at all.
  2. The Grid slows upon hitting the Cube's top face, and stops fully on the bottom face.
  3. The Grid stops at the top face, but two opposite Cube corners are visible.
  4. ...no perceptible difference.
  5. Finally a working sim. This is good, because it is the maximum step level.
- Surprisingly, the more vertices you have in your hit object, the less likely the Soft Body is to react with it. If you try letting it hit a Plane, it might stop, but a subdivided Grid might fail.

Sensors

Soft bodies do not work with the Collision, Touch, Near, and Radar logic brick sensors.

Goal Weights

TODO:
http://www.blender.org/documentation/blender_python_api_2_62_release/bpy.ops.curve.html#bpy.ops.curve.spline_weight_set

Force Fields

A common practice within the non-BGE Cloth simulator is to employ Force Fields to animate the cloth.

These do not work in the BGE, so you will have to figure out a way to use Python (or perhaps plain Logic Bricks) to apply forces to the Soft Body objects.

All Types

Summary - No Collision | Static | Dynamic | Rigid Body | Soft Body | Occlude | Sensor | Navigation Mesh | Character | Vehicle

Vehicle Controller

Intro

The **Vehicle Controller** is a special type of physics object that the Physics Engine (bullet) recognizes.

It is composed of a **rigid body** representing the chassis and a set of wheels that are set to **no collision**. Emphasizing the distinction between a GameEngine/Logical/Render object and its representation for the Physics Engine is important.

To simulate a vehicle as a true rigid body, on top of also rigid body wheels, with a real suspension system made with joints, would be far too complicated and unstable. Cars and other vehicles are complicated mechanical devices and most often we do not want to simulate that, only that it 'acts as expected'. The Vehicle Controller exists to provide a dedicated way of simulating a vehicle behavior without having to simulate all the physics that would actually happen in the real world. It abstracts the complexity away by providing a simple interface with tweakable parameters such as suspension force, damping and compression.

How it works

Bullet's approach to a vehicle controller is called a '**Raycast Vehicle'**. Collision detection for the wheels is approximated by ray casts and the tire friction is an anisotropic friction model.

A raycast vehicle works by casting a ray for each wheel. Using the ray's intersection point, we can calculate the suspension length and hence the suspension force that is then applied to the chassis, keeping it from hitting the ground. In effect, the vehicle chassis 'floats' along on the rays.

The friction force is calculated for each wheel where the ray contacts the ground. This is applied as a sideways and forwards force.

You can check Kester Maddock's approach to vehicle simulation here. It includes some common problems, workarounds and tips and tricks.

How to Use

Currently the Vehicle Controller can only be used as a constraint via Python. There are plans to add it to the interface.

Setup

You should have a body acting as the chassis, set it as a 'Rigid Body'.
The wheels should be separate objects set to 'No Collision'. The vehicle controller will calculate the collisions for you as rays so, if you set it to something else, it will calculate it twice in different ways and produce weird results.

Collisions

A cylinder is typically a good collision shape for the wheels. For the chassis, the shape should be rough, like a box. If the vehicle is very complicated, you should split it into simpler objects and parent those (with their collision shapes) to the vehicle controller so that they will follow it. If your vehicle even has moving bits (weapons, wrecking balls, trolleys etc) they should also be simulated separately and connected to the vehicle as a joint.

Python

Assembling the Vehicle

The overall steps are:

- create a constraint for the vehicle and save its ID for future reference
- attach the wheels
- set wheel parameters: influence, stiffness, damping, compression and friction
- init variables

You can see an example in the file below.

Controlling the Vehicle

This is done in 2 parts and it should be modeled according to the desired behavior. You should think of your gameplay and research appropriate functions for the input. For instance, can the vehicle reverse? jump? drift? does it turn slowly? How much time does it take to brake or get to full speed? The first part is **response to keys**. Whenever the player presses a key, you should set a value accordingly, such as increase acceleration. Example:

```
if   key[0] == events.UPARROWKEY:
    logic.car["force"]  = -15.0
elif key[0] == events.RIGHTARROWKEY:
    logic.car["steer"] -= 0.05
```

The second part is to **compute the movement** according to your functions.

```
## apply engine force ##
for i in range(0, totalWheels):
    vehicle.applyEngineForce(logic.car["force"],i)
...
## slowly ease off gas and center steering ##
logic.car["steer"] *= 0.6
```

```
   logic.car["force"] *= 0.9
```

Both should be run each frame.

Example

demo_file.zip (last update 9 September 2014)

Sensor

The object detects static and dynamic objects but not other collisions sensors objects. The Sensor is similar to the physics objects that underlie the Near and Radar sensors. Like the Near and Radar object it is:

- static and ghost
- invisible by default
- always active to ensure correct collision detection
- capable of detecting both static and dynamic objects
- ignoring collision with their parent
- capable of broadphase filtering based on:
    - Actor option: the collisioning object must have the Actor flag set to be detected
    - property/material: as specified in the collision sensors attached to it.

Broadphase filtering is important for performance reason: the collision points will be computed only for the objects that pass the broadphase filter.

- automatically removed from the simulation when no collision sensor is active on it

Unlike the Near and Radar object it can:

- take any shape, including triangle mesh
- be made visible for debugging (just use the Visible actuator)
- have multiple collision sensors using it

Other than that, the sensor objects are ordinary objects. You can move them freely or parent them. When parented to a dynamic object, they can provide advanced collision control to this object.

The type of collision capability depends on the shape:

- box, sphere, cylinder, cone, convex hull provide volume detection.
- triangle mesh provides surface detection but you can give some volume to the surface by increasing the margin in the Advanced Settings panel. The margin applies on both sides of the surface.

Performance tip:

- Sensor objects perform better than Near and Radar: they do less synchronizations because of the Scenegraph optimizations and they can have multiple collision sensors on them (with different property filtering for example).
- Always prefer simple shape (box, sphere) to complex shape whenever possible.
- Always use broadphase filtering (avoid collision sensor with empty propery/material)
- Use collision sensor only when you need them. When no collision sensor is active on the sensor object, it is removed from the simulation and consume no CPU.

Known limitations:

- When running Blender in debug mode, you will see one warning line of the console:
  "warning btCollisionDispatcher::needsCollision: static-static collision!"
  In release mode this message is not printed.
- Collision margin has no effect on sphere, cone and cylinder shape.

Settings

Invisible
    See Here

Collision Bounds

See Here.

Occlude Object Type

If an Occlude type object is between the camera and another object, that other object will not be rasterized (calculated for rendering). It is "culled" because it is "occluded".

The overall process (also known as "o-culling") disregards, by default, anything outside of the View Frustum, meaning you don't have to worry about anything outside the view's rectangular border. Inside this border, you might want to do additional culling.

In this demo .blend, BGE-Physics-Objects-Occluder.blend, you will see:

- A messed-up, subdivided Cube named "Cube".
- Another one behind a "Physics Type: Occlude" plane, named "Cube.BG".
- Another one outside the view Frustum, named "Cube.OffCamera".

Now observe what happens to the profiling stats for each of the following (in order):

1. Hit P as the scene is. It hums along at a fairly slow rate. On my system the Rasterizer step takes 130ms. The framerate will finally jump up once the "Cube" object has completely moved out of the view frustum. ??? - It's as if the Occluder doesn't do anything while the Cube is behind it.
2. Delete the "Cube.OffCamera" object above, and notice that there is no improvement in speed. This is the view frustum culling working for you - it does not matter if that object exists or not.
3. Hit Z to view wireframe. Notice that in the 3D Viewport you can see "Cube.BG", but once you hit P, it is not there.
4. Make the "Occluder" object take up the whole camera's view with SX5. You will see a huge leap in framerate, since almost nothing is being Rasterized. On my system the Rasterizer step drops to 5ms.
5. Try a run with World properties » Physics » Occlusion Culling disabled. It will be slow again.
6. Reenable World properties » Physics » Occlusion Culling and run it one more time to prove to yourself that your speed is back.
7. Change the Occluder to "Physics Type: Static". Notice that it is back to the original slowness.
8. Change it back to "Physics Type: Occlude".
9. Now make the "Occluder" invisible. The framerate is back down to its original, slow rate. ??? - I thought this was supposed to work when invisible.

TODO

Incorporate some of the 2.49 Release Notes Details.

Details

As far as Physics is concerned, this type is equivalent to Rigid Object "No collision". The reason why the Occluder mode is mutually exclusive with other physics mode is to emphasize the fact that occluders should be specifically designed for that purpose and not every mesh should be an occluder. However, you can enable the Occlusion capability on physics objects using Python and Logic bricks - see (Link- TODO)

When an occluder object enters the view frustrum, the BGE builds a ZDepth buffer from the faces of that object. Whether the faces are one-side or two-side is important: only the front faces and two-side faces are used to build the ZDepth buffer. If multiple occluders are in the view frustrum, the BGE combines them and keeps the most foreground faces.

The resolution of the ZDepth buffer is controllable in the World settings with the "Occlu Res" button:

By default the resolution is 128 pixels for the largest dimension of the viewport while the resolution of the other dimension is set proportionally. Although 128 is a very low resolution, it is sufficient for the purpose of culling. The resolution can be increased to maximum 1024 but at great CPU expense.

The BGE traverses the DBVT (Dynamic Bounding Volume Tree) and for each node checks if it is entirely hidden by the occluders and if so, culls the node (and all the objects it contains).

To further optimize the feature, the BGE builds and uses the ZDepth buffer only when at least one occluder is in the view frustrum. Until then, there is no performance decrease compared to regular view frustrum culling.

Recommendations

Occlusion culling is most useful when the occluders are large objects (buildings, mountains, ...) that hide many complex objects in an unpredictable way. However, don't be too concerned about performance: even if you use it inappropriately, the performance decrease will be limited due to the structure of the algorithm.

There are situations where occlusion culling will not bring any benefit:

- If the occluders are small and don't hide many objects.
  - In that case, occlusion culling is just dragging your CPU down).
- If the occluders are large but hides simple objects.
  - In that case you're better off sending the objects to the GPU).
- If the occluders are large and hides many complex objects but in a very predictable way.
  - Example: a house full of complex objects. Although occlusion culling will perform well in this case, you will get better performance by implementing a specific logic that hides/unhides the objects; for instance making the objects visible only when the camera enters the house).
- Occluders can be visible graphic objects but beware that too many faces will make the ZDepth buffer creation slow.
  - For example, a terrain is not a good candidate for occlusion: too many faces and too many overlap. Occluder can be

invisible objects placed inside more complex objects (ex: "in the walls" of a building with complex architecture). Occluders can have "holes" through which you will see objects.

Performance

When developing games, game engineers, software and hardware developers uses some tools to fine tune their games to specific platforms and operating systems, defining a basic usage scenario whereas the users would have the best possible experience with the game.

Most of these tools, are software tools available for the specific Game Engines whereas the games were being developed and will run.

Blender Game Engine also comes with some visual tools to fine tune the games being developed, so the game developers could test the best usage scenario and minimum software and hardware requirements to run the game.

In Blender, those tools are available at the System and Display tab of Render Context in the Properties Window. There are options for specific performance adjusts and measurements, ways to control the frame rate or the way the contents are rendered in Blender window (game viewport) while the game runs, as well as controls for maintainnig geometry allocated in graphic cards memory.

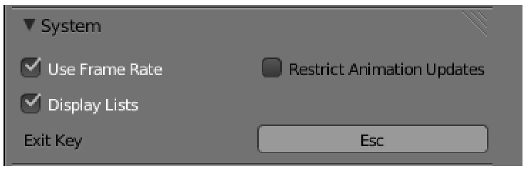
Blender Game Engine rendering system controls
: System - Controls for Scene rendering while the game is running.

Blender Game Engine Performance measurements
: Display - Controls for showing specific data about performance while the game is running.

System

The System tab at the Render context of the Properties Window, let the game developer specify options about the system performance regarding to frame discards and restrictions about frame renderings, the key to stop the Blender Game Engine, and whether to maintain geometry in the internal memory of the Graphic card.

Options



System tab at the Render Context

Use Frame Rate
When checked, this will inform Blender whether to run freely without frame rate restrictions or not. The frame rate is specified at the Display tab of the Render Context of the Properties Window. For more information about frame rates, see the Display page.

Display Lists
When checked, this will tell Blender to maintain the lists of the meshes geometry allocated at the GPU memory. This can help to speed up viewport rendering during the game if you have enough GPU memory to allocate geometry and textures.

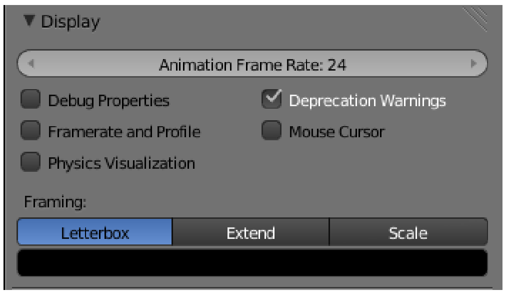
Restrict Animation Updates
When checked, this will force Blender game engine to discard frames (even at the middle of redrawing, sometimes causing *tearing* artifacts) if the rate of frame rendered by the GPU is greater than the specified at the Display Tab.

Exit Key
Clicking at this button will ask the user to type a key to specify a key to stop the game engine from running.

Display

The Display tab at the Render context of the Properties Window, let the game developer specify the maximum frame rate of the animations shown during the game execution, whether to see informations like framerate and profile, debug properties, physics geometry visualization, warnings, if the mouse cursor is shown during the game execution, and options to specify the framing style of the game to fit the window with the specified resolution.

Options


▼ Display

Animation Frame Rate: 24

- [ ] Debug Properties
- [x] Deprecation Warnings
- [ ] Framerate and Profile
- [ ] Mouse Cursor
- [ ] Physics Visualization

Framing: [✓ Letterbox] Extend Scale

Fig. 1 - Display Tab at the Render Context

Animation Frame Rate
: This numeric field/slider specify the maximum frame rate at which the game will run.

Minimum is **1**, maximum is **120**.

Debug Properties
: When checked, if a property was previously checked to be debugged during the game, the values of this property will be shown with the Framerate and Profile contents.

Framerate and Profile
: When checked, this will show values for each of the calculations Blender is doing while the game is running, plus the properties marked to be debugged. Each of the values are explained at the Framerate and Profile page.

Physics visualization
: Shows a visualization of phisycs bounds and interactions (like hulls and collision shapes), and their interaction.

Deprecation Warnings
: Every time when the game developer uses a deprecated functionality (which in some cases are outdated or crippled OpenGL Graphic cards functions), the system will emit warnings about the deprecated function.

Mouse Cursor
: Wether to show or not the mouse cursor when the game is running.

Framing

There are three types of framing available for the Blender Game Engine, *Letterbox*, *Extend* and *Scale*

Letterbox
: Show the entire viewport of the game in display window, using horizontal and/or vertical bars when needed.

Extend
: Show the entire viewport of the game in display window, viewing more horizontally or vertically.

Scale
: Stretch or Squeeze the viewport to fill the display window.

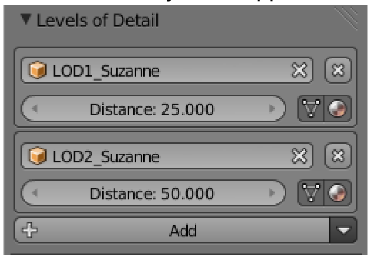
Color Bar
: This will let the game developer choose the bar colors when using the **Letterbox** Framing mode.

Introduction

When creating visual assets it is often desirable to have a high amount of detail in the asset for up close viewing. However, this high amount of detail is wasted if the object is viewed from a distance, and brings down the scene's performance. To solve this, the asset can be swapped out at certain viewing distances. This is commonly referred to as a level of detail system. Each visual step of the asset is known as a level of detail. Levels of detail are most appropriate to use when you have a large scene where certain objects can be viewed both up close and from a distance.

Settings

Modifiers on Level of Detail Objects
Any level of detail objects that have a modifier do not display correctly in the game engine. You will need to apply any modifiers for level of detail objects to appear correctly. A fix for this is being looked into.



Level of detail settings can be found in the Object settings when the renderer is set to Blender Game. In the Levels of Detail panel is a button to add a new level of detail to the current object. The settings for each level of detail is displayed in its own box. The exception to this is the base level of detail. This is automatically setup as the current object with a distance setting of 0. To remove a level of detail, click on the X button in the top right corner of the box of the level to be removed.

Object
: The object to use for this level of detail.

Distance
: The distance at which this level of detail becomes visible.

Use Mesh
: When this option is enabled, the mesh from the level of detail object is used until a lower level of detail overrides it.

Use Material
: When this option is enabled, the material from the level of detail object is used until a lower level of detail overrides it.

Tools

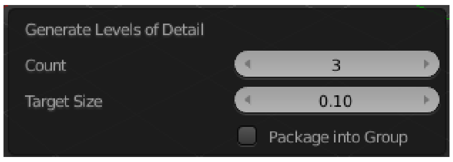
Some tools for making levels of detail easier to manage and create can be found from the drop down menu next to the add button in the Levels of Detail panel.

Set By Name

Searches the scene for specifically named objects and attempts to set them up as levels of detail on the currently selected object. The selected object must be the base level of detail (e.g. LOD0). This can be useful to quickly setup levels of detail on imported assets. In order to make use of this tool, your naming must be consistent, and each level must be prefixed or suffixed with "lodx" where x is the level that object is intended for. The case on "lod" must be consistent across all objects. Below are some example names that the tool will recognize.

- LOD0_Box, LOD1_Box, LOD2_Box
- Box.lod0, Box.lod1, Box.lod2
- LoD0box, LoD1box, LoD2box

Generate


Generate Levels of Detail
Count 3
Target Size 0.10
Package into Group

This tool generates and sets up levels of details based on the selected object. Generation is done using the decimate modifier. Generation does not apply the modifier to allow further changing the settings. Generated objects are automatically named based on the level they are generated for. Below are some settings for the operator.

Count
: The number of levels desired after generation. This operator creates Count - 1 new objects.

Target Size
: The ratio setting for the decimate modifier on the last level of detail. The ratio settings for the other levels is determined by linear interpolation.

Package into Group
: With this setting enabled the operator performs some extra tasks to make the asset ready for easy linking into a new file. The base object and all of its levels of detail are placed into a group based on the base object's name. Levels other than the base are hidden for both the viewport and rendering. This simplifies the appearance of the system and does not affect the

appearance of the base object. Finally, all levels are parented to the base object to remove clutter from the outliner.

Clear All

Clears the level of detail settings from the current object.

Python API

This site is currently under development.

To see the full Python API please click on the following link Python API

More informations:

- Bullet physics
- Video Texture

Bullet physics Python API

Bullet Physics provides collision detection and rigid body dynamics for the Blender Game Engine. It takes some settings from Blender that previously were designed for the former collision detection system (called Sumo).

However, new features don't have an user interface yet, so Python can be used to fill the gap for now.

Features:

- Vehicle simulation.
- Rigid body constraints: hinge and point to point (ball socket).
- Access to internal physics settings, like deactivation time, debugging features.

Easiest is to look at the Bullet physics demos, how to use them. More information can be found here.

Python script example:

```
import PhysicsConstraints
print dir(PhysicsConstraints)
```

Note about parameter settings
Since this API is not well documented, it can be unclear what kind of values to use for setting parameters. In general, damping settings should be in the range of 0 to 1 and stiffness settings should not be much higher than about 10.

The VideoTexture module: bge.texture

The `VideoTexture` module allows you to manipulate textures during the game. Several sources for texture are possible: video files, image files, video capture, memory buffer, camera render or a mix of that. The video and image files can be loaded from the internet using an URL instead of a file name. In addition, you can apply filters on the images before sending them to the GPU, allowing video effect: blue screen, color band, gray, normal map. `VideoTexture` uses FFmpeg to load images and videos. All the formats and codecs that FFmpeg supports are supported by `VideoTexture`, including but not limited to:

- AVI
- Ogg
- Xvid
- Theora
- dv1394 camera
- video4linux capture card (this includes many webcams)
- videoForWindows capture card (this includes many webcams)
- JPG

Changes to VideoTexture in Blender 2.6

The `VideoTexture` module is now simply called `bge.texture`.

How it works

The principle is simple: first you identify an existing texture by object and name, then you create a new texture with dynamic content and swap the two textures in the GPU. The GE is not aware of the substitution and continues to display the object as always, except that you are now in control of the texture. At the end, the new texture is deleted and the old texture restored.

The present page is a guide to the `VideoTexture` module with simple examples.

Game preparation

Before you can use the thing `VideoTexture` module, you must have objects with textures applied appropriately.

Imagine you want to have a television showing live broadcast programs in the game. You will create a television object and UV-apply a different texture at the place of the screen, for example "`tv.png`". What this texture looks like is not important; probably you want to make it dark grey to simulate power-off state. When the television must be turned on, you create a dynamic texture from a video capture card and use it instead of `tv.png`: the TV screen will come to life.

You have two ways to define textures that `VideoTexture` can grab:

1. Simple UV texture.
2. Blender material with image texture channel.

Because `VideoTexture` works at texture level, it is compatible with all GE fancy texturing features: GLSL, multi-texture, custom shaders, etc.

First example

Let's assume that we have a game object with one or more faces assigned to a material/image on which we want to display a video.

The first step is to create a `Texture` object. We will do it in a script that runs once. It can be at the start of the game, the video is only played when you refresh the texture; we'll come to that later. The script is normally attached to the object on which we want to display the video so that we can easily retrieve the object reference:

```
import VideoTexture

contr = GameLogic.getCurrentController()
obj = contr.owner

if not hasattr(GameLogic, 'video'):
```

The check on "`video`" attribute is just a trick to make sure we create the texture only once.

Find material

```
    matID = VideoTexture.materialID(obj, 'IMvideo.png')
```

`VideoTexture.materialID()` is a handy function to retrieve the object material that is using `video.png` as texture. This method will work with Blender material and UV texture. In case of UV texture, it grabs the internal material corresponding to the faces that are assigned to this texture. In case of Blender material, it grabs the material that has an image texture channel matching the name as first channel.

The "`IM`" prefix indicates that we're searching for a texture name but we can also search for a material by giving the "`MA`" prefix. For example, if we want to find the material called `VideoMat` on this object, the code becomes:

```
    matID = VideoTexture.materialID(obj, 'MAVideoMat')
```

Create texture

`VideoTexture.Texture` is the class that creates the `Texture` object that loads the dynamic texture on the GPU. The constructor takes one mandatory and three optional arguments:

`gameObj`
: The game object.

`materialID`
: Material index as returned by `VideoTexture.materialID()`, 0 = first material by default.

`textureID`
: Texture index in case of multi-texture channel, 0 = first channel by default.
In case of UV texture, this parameter should always be 0.

`textureObj`
: Reference to another `Texture` object of which we want to reuse the texture.
If we use this argument, we should not create any source on this texture and there is no need to refresh it either: the other `Texture` object will provide the texture for both materials/textures.

```
GameLogic.video = VideoTexture.Texture(obj, matID)
```

Make texture persistent

Note that we have assigned the object to a `GameLogic` “`video`” attribute that we create for the occasion. The reason is that the `Texture` object must be persistent across the game scripts. A local variable would be deleted at the end of the script and the GPU texture deleted at the same time. `GameLogic` module object is a handy place to store persistent objects.

Create a source

Now we have a `Texture` object but it can’t do anything because it does not have any source. We must create a source object from one of the possible sources available in `VideoTexture`:

`VideoFFmpeg`
: Moving pictures.
Video file, video capture, video streaming.

`ImageFFmpeg`
: Still pictures.
Image file, image on web.

`ImageBuff`
: Image from application memory.
For computer generated images, drawing applications.

`ImageViewport`
: Part or whole of the viewport (=rendering of the active camera displayed on screen).

`ImageRender`
: Render of a non active camera.

`ImageMix`
: A mix of 2 or more of the above sources.

In this example we use a simple video file as source. The `VideoFFmpeg` constructor takes a file name as argument. To avoid any confusion with the location of the file, we will use `GameLogic.expandPath()` to build an absolute file name, assuming the video file is in the same directory as the blend file:

```
movie = GameLogic.expandPath('//trailer_400p.ogg')
GameLogic.video.source = VideoTexture.VideoFFmpeg(movie)
```

We create the video source object and assign it to the `Texture` object `source` attribute to set the source and make it persistent: as the `Texture` object is persistent, the source object will also be persistent.

Note that we can change the `Texture` source at any time. Suppose we want to switch between two movies during the game. We can do the following:

```
GameLogic.mySources[0] = VideoTexture.VideoFFmpeg('movie1.avi')
GameLogic.mySources[1] = VideoTexture.VideoFFmpeg('movie2.avi')
```

And then assign (and reassign) the source during the game:

```
GameLogic.video.source = GameLogic.mySources[movieSel]
```

Setup the source

The `VideoFFmpeg` source has several attributes to control the movie playback:

`range`
: [start,stop] (*floats*).
Set the start and stop time of the video playback, expressed in seconds from beginning. By default the entire video.

`repeat`
: (*integer*).
Number of video replay, -1 for infinite.

`framerate`
: (*float*).
Relative frame rate, <1.0 for slow, >1.0 for fast.

`scale`
: (*bool*).
Set to True to activate fast nearest neighbour scaling algorithm.
Texture width and height must be a power of 2. If the video picture size is not a power of 2, rescaling is required. By default `VideoTexture` uses the precise but slow `gluScaleImage()` function. Best is to rescale the video offline so that no scaling is necessary at runtime!

`flip`
: (*bool*).
Set to True if the image must be vertically flipped.
FFmpeg always delivers the image upside down, so this attribute is set to True by default.

`filter`
: Set additional filter on the video before sending to GPU.
Assign to one of `VideoTexture` filter object. By default the image is send unchanged to the GPU. If an alpha channel is present in the video, it is automatically loaded and sent to the GPU as well.

We will simply set the `scale` attribute to True because the `gluScaleImage()` is really too slow for real time video. In case the video dimensions are already a power of 2, it has no effect.

```
GameLogic.video.source.scale = True
```

Play the video

We are now ready to play the video:

```
GameLogic.video.source.play()
```

Video playback is not a background process: it happens only when we refresh the texture. So we must have another script that runs on every frame and calls the `refresh()` method of the `Texture` object:

```
if hasattr(GameLogic, 'video'):
    GameLogic.video.refresh(True)
```

If the video source is stopped, `refresh()` has no effect. The argument of `refresh()` is a flag that indicates if the texture should be recalculated on next refresh. For video playback, you definitively want to set it to True.

Checking video status

Video source classes (such as VideoFFMpeg) have an attribute `status`. If video is playing, its value is 2, if it's stopped, it's 3. So in our example:

```
if GameLogic.video.source.status == 3:
    #video has stopped
```

Advanced work flow

True argument in `Texture.refresh()` method simply invalidates the image buffer after sending it to the GPU so that on next frame, a new image will be loaded from the source. It has the side effect of making the image unavailable to Python. You can also do it manually by calling the `refresh()` method of the source directly.

Here are some possible advanced work flow:

- Use the image buffer in python (doesn't effect the Texture):

  ```
  GameLogic.video.refresh(False)
  image = GameLogic.video.source.image
  # image is a binary string buffer of row major RGBA pixels
  # ... use image
  # invalidates it for next frame
  GameLogic.video.source.refresh()
  ```

- Load image from source for python processing wihtout download to GPU:

```
# note that we don't even call refresh on the Texture
```

```
    # we could also just create a source object without a Texture object

     image = GameLogic.video.source.image
     # ... use image
     GameLogic.video.source.refresh()
```

- If you have more than 1 material on the mesh and you want to modify a texture of one particular material, get its ID

```
matID=VideoTexture.materialID(gameobj,"MAmat.001")
```

GLSL material can have more than 1 texture channel, identify the texture by the texture slot where it is defined, here 2

```
tex=VideoTexture.Texture(gameobj, matID, 2)
```

Advanced demos

Here is a demo that demonstrates the use of two videos alternatively on the same texture. Note that it requires an additional video file which is the elephant dream teaser. You can replace with another other file that you want to run the demo.

Here is a demo that demonstrates the use of the `ImageMix` source. `ImageMix` is a source that needs sources, which can be any other `Texture` source, like `VideoFFmpeg`, `ImageFFmpeg` or `ImageRender`. You set them with `setSource()` and their relative weight with `setWeight()`. Pay attention that the weight is a short number between 0 and 255, and that the sum of all weights should be 255. `ImageMix` makes a mix of all the sources according to their weights. The sources must all have the same image size (after reduction to the nearest power of 2 dimension). If they don’t, you get a Python error on the console.

Standalone Player

The standalone player allows a Blender game to be run without having to load the Blender system. This allows games to be distributed to other users, without their requiring a detailed knowledge of Blender (and also without the possibility of unauthorised modification). Note that the Game Engine Save as Runtime is an addon facility which must be pre-loaded before use.

The following procedure will give a standalone version of a working game.

- **File - User Preferences - Addons: - Game Engine - Save as Game Engine Runtime - Install Addon**(button).

(You can also **Save as Default** button, in which case the add-on will always be present whenever Blender is re-loaded).

- **File - Export - Save as Game Engine Runtime - (give appropriate directory/filename)- Save as Game Engine Runtime** (button).

The game can then be executed by running the appropriate .exe file. Note that all appropriate libraries are automatically loaded by the add-on.

If you are interested in licensing your game, read Licensing for a discussion of the issues involved.

Exporting…

If the game is to be exported to other computers, make a new empty directory for the game runtime and all its ancilliary libraries etc. Then make sure the **whole directory** is transferred to the target computer

Licensing of Blender Games

The licensing of games created for distribution using the Blender Game Engine and the Standalone Player is complicated by Blender's status as open-source software. This page aims to describe the problems, and present some possible solutions.

Blender is distributed as open-source software distributed and owned by the Blender Foundation under the GNU General Public License (GPL). In brief, while the Blender system itself is available to everyone, you own anything that you make using Blender (scripts, texture, rendered artwork etc.). See http://www.blender.org/education-help/faq/gpl-for-artists/ for further details.

Standalone Player License

Unfortunately, this does not extend to games or other artwork distributed to run under the Blender Standalone Player. To distribute your game you need to create an executable (run time). What it does is take your Blender .blend file and put it "inside" the Standalone Player - a stripped-down version of Blender containing only the functions corresponding to the Blender Game Engine. The resulting executable file falls into the category of "derivatives" of the original program (i.e. a hybrid of your file with the Standalone Player itself), and therefore must be licensed as GPL.

Distributing Games

There are possible solutions to the problem of how to distribute your game with suitable license protection:

1) Do not protect your Blender Game by license. Are you really sure that you need to license it? Remember the old adage "Imitation is the sincerest form of flattery".

2) Use the Game Actuator, which enables a basic .blend game file to start. By making a basic file which contains an "Always" sensor to run, and allowing this to activate a "Game" actuator to load and run the full content of your game, this gets round the problem. Your main file is now "outside" the Standalone Player, so that it need not be open to GPL and is therefore "legally protected". Although your game is not fully protected with this system, it affords a similar level of protection to that used in most other distributed games. The fact that others can access your .blend file does not mean that it can be used for purposes not covered by the license you want.

(Acknowledgements: This page is based on information contained in the blog file of Dalai Felinto).

Page status (reviewing guidelines)

Text This sub-section needs some review & formatting…
Proposed fixes: none

- Game dev guide for Android
- Building Blender for Android
- Building Blender with GLES